Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук_____
(повна назва)

Кафедра Програмної інженерії_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти – другий (магістерський)

Дослідження та порівняльний аналіз алгоритмів управління багтрекінговими системами

(тема)

Виконав: студент 2 курсу, групи ІПЗм-18-3
_____Чернявський М.А._____
(прізвище, ініціали)

____Освітньо-наукова програма___
(тип програми)

___Інженерія програмного забезпечення___
(повна назва освітньої програми)

Керівник_ доц. каф. ПІ Лановий О.Ф._
(посада, прізвище, ініціали)

Допускається до захисту

Зав. Кафедри ПІ, проф. _____ З.В.Дудар

20__р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет _Комп'ютерних наук_____

Кафедра _Програмної інженерії_____

Рівень вищої освіти – другий (магістерський)

Спеціальність _121 – Інженерія програмного забезпечення_____
(код і повна назва)

Тип програми _освітньо-наукова програма_____

Освітня програма _Інженерія програмного забезпечення_____

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____20__ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові _Чернявському Миколі Анатолійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи _Дослідження та порівняльний аналіз алгоритмів управління багтрекінговими системами
затверджена наказом університету від "__"_____ 20__ р. № _____
заповнюється вручну після отримання наказу
2. Термін подання студентом роботи до екзаменаційної комісії
05 червня 2020 р.
3. Вихідні дані до роботи:проект, баг, виявлення дефектів, керування процесом виявлення багів, керування проектами
4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, загальні правила роботи з багтрекінговими системами, аналіз існуючих систем, класифікація та порівняльний аналіз алгоритмів систем, вибір оптимальної системи.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Аналіз предметної галузі	02.04.2020	
2.	Огляд існуючих методів адаптації домену	08.04.2020	
3.	Методи семантичної сегментації	10.04.2020	
4.	Підготовка пояснювальної записки	30.04.2020	
5.	Спецчастина	01.05.2020	
6.	Підготовка презентацій та доповіді	03.05.2020	
7.	Попередній захист	07.05.2020	
8.	Нормоконтроль, рецензування	08.05.2020	
9.	Занесення диплома в електронний архів	09.05.2020	
10.	Допуск до захисту у зав. кафедри	10.05.2020	

Дата видачі завдання 30.03. 2020 р.

Студент _____
(підпис)

Керівник роботи _____ ____доц. каф. ПІ Лановий О.Ф,.__
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної магістерської роботи містить 54 сторінки, 16 рисунків, 2 таблиці.

ПРОЕКТ, БАГ, ВИЯВЛЕННЯ ДИФЕКТІВ, КЕРУВАННЯ ПРОЦЕСОМ ВИЯВЛЕННЯ БАГІВ, КЕРУВАННЯ ПРОЕКТАМИ, БАГТРЕКІНГОВА СИСТЕМА

Об’єкт дослідження – алгоритми управління багтрекінговими системами у проектах.

Мета дослідження – аналіз існуючих алгоритмів управління багтрекінговими системами, та виявлення найбільш оптимального алгоритму для вибору системи для долучення до проекту.

В результаті роботи проведеного дослідження було виконано аналіз та порівняння існуючих багтрекінгових систем, та створено алгоритм вибору оптимальної системи для кожного проекту.

PROJECT, BUG, DIFFECTION DETECTION, BUG DEVELOPMENT PROCESS CONTROL, PROJECT MANAGEMENT, BUGTRAKING SYSTEM.

The object of research is algorithms for managing bug-tracking systems in projects.

The purpose of the study is to analyze the existing algorithms for managing bug-tracking systems, and to identify the most optimal algorithm for selecting a system to join the project.

As a result of the research, an analysis and comparison of existing bug-tracking systems was performed, and an algorithm for selecting the optimal system for each project was created.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

СУБД – Система управления баз данных

ПЗ – програмне забезпечення

BTT – Bug tracking tool

БД – база даних

RSS – Rich Site Summary

WF – Workflow

SLA – Service Level Agreement

ЖЦПЗ – життєвий цикл програмного забезпечення

QA –quality assurance

ВСТУП

Нам всім відома популярна приказка «Людині властиво помилятися». Основна мета розробників програмного забезпечення - дізнатися, яка помилка була допущена, виправити її і потім витягти уроки, щоб не допускати подібних помилок надалі. Додатки, для розробки програмного забезпечення, надають широкий спектр інструментів і методологій, які допомагають керувати різними етапами ЖЦПЗ. Ці додатки допомагають в процесі розробки ПЗ відстежувати виникнення помилок, відстежувати версії, контролювати хід та етапи розробки, а так само інші дії реалізації ПЗ.

Коли кількість помилок мінімально, відсутні специфічні вимоги до інструментів їх відстеження. Але коли кількість помилок стає високим, або вимоги до надійності продукту (відсутність помилок) критичне - виникає гостра необхідність у використанні спеціальних програмних засобів. Bug Tracking Tools – програмне забезпечення, яке призначене для допомоги у відстеженні помилок в програмних продуктах. Можливо використовувати он-лайн таблицями (такими як Google excel), та це буде не зручно.

В наш час, коли програмні продукти стали надзвичайно великими, для задоволення різноманітних потреб користувачів, використання таких систем особливо актуальні. Діяльність, по супроводу в життєвому циклі розробки програмного забезпечення, використовує велику частину бюджету проекту, пов'язаних з витратами на розробку програмного забезпечення.

Отже, відстеження помилок має величезне значення, як частина супроводу і обслуговування програмного забезпечення. Існують різні інструменти для пошуку дефектів (багів) і їх відстеження. Наприклад програмні продукти, які використовують програми з відкритим вихідним кодом. У розробки програмного забезпечення з відкритим вихідним кодом, є велика потреба в системах стеження за вадами тому, що члени команди розробників можуть бути поширені по всьому світу, і мати різні часові пояси, і не мати можливості телефонувати або зустрічатися особисто[7]. У таких випадках допомагають трекери помилок для

відстеження повідомлень про проблеми, а також допомагає в управлінні конфігурацією програмного забезпечення.

Системи відстеження помилок мають безліч переваг, отже, стають все більш популярними серед організацій по розробці програмного забезпечення.

Нижче перераховані деякі важливі переваги систем стеження за вадами:

- виявлення помилок додатків;
- збільшує прозорість процесу розробки;
- покращує відстеження помилок і дозволів;
- допомагає в управлінні помилками для майбутніх випусків;
- допомога в плануванні релізів;
- управління планування ресурсів;
- приоритезация помилок при прийнятті рішень по SLA;
- чи відстежує статус помилок;
- допомагає в обміні і консолідації інформації, щоб допомогти конфігурації програмного забезпечення;
- підвищує продуктивність програмного забезпечення, за рахунок підвищення якості програмного забезпечення.

Основною метою дослідження є аналіз існуючих багтрекінгових систем, та виявлення найбільш оптимального алгоритму долучення їх до проекту.

1АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Методи управління у програмних проектах

За усю історію людства, було сплановано та реалізовано велика кількість проектів, від побудування Пірамід у Гізі, до відправлення людини на Місяць. Усі найсміливіші людські починання вимагали узгодженої роботи тисячі людей. А це передбачає складну систему управління проектами.

Усі проекти різні. Не існує єдиної ідеальної системи управління проектами, яка підходить під будь який тип задач. Також не існує системи, котра підійшла би кожному керівнику та була зручної для кожного в команді. За весь час було створено велику кількість методів управління проектами.

Існує багато методів управління в програмних проектах, такі як:

- класичний проектний менеджмент;
- Agile;
- Scrum;
- Lean;
- Six Sigma;
- PRINCE2;
- Kanban та інші.

Розглянемо декілька найбільш популярних з них.

1.1.1 Класичне проектне управління.

Найбільш очевидний спосіб зробити свій проект біль керованим, це розбити увесь процес його виконання на окремі та послідовні етапи. Саме на такому методі базується традиційне проектне управління (також відоме як каскадна модель, або водоспадна). Схема робочого процесу (рис 1.1).

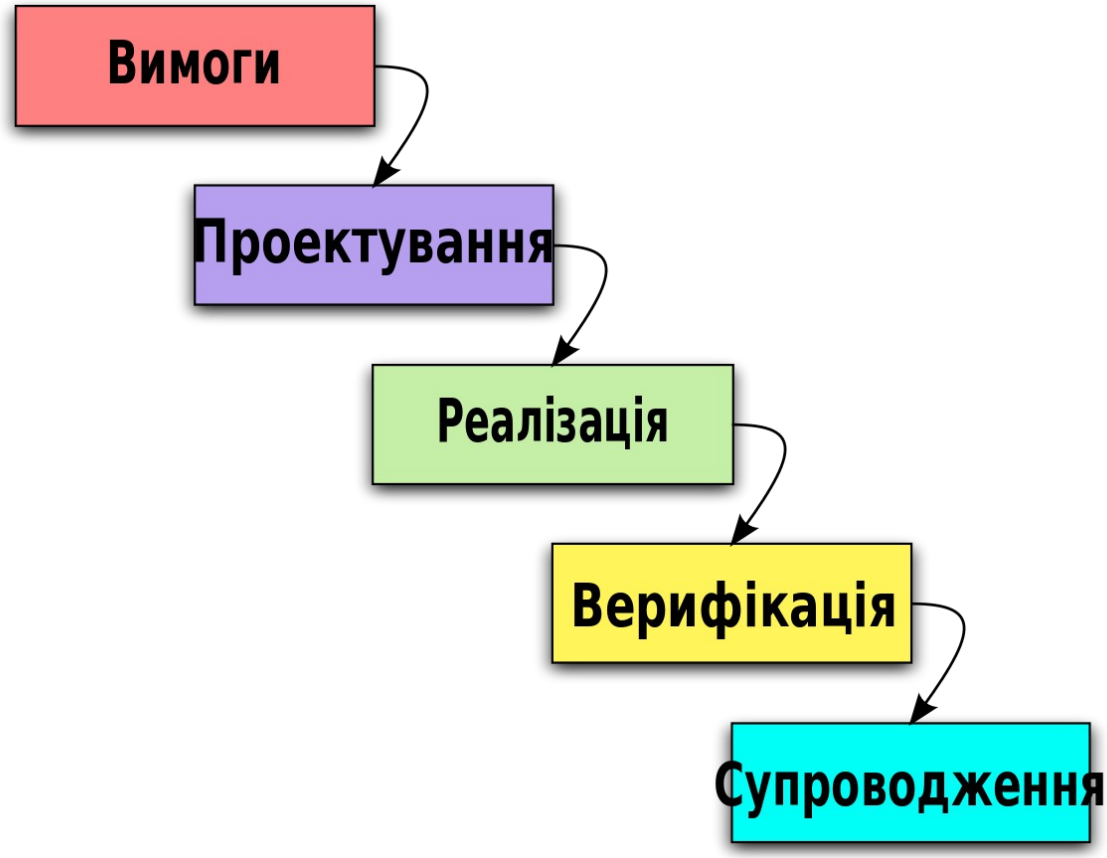


Рис 1.1 – традиційна модель

Розробка програмного продукту за класичною моделлю складається з п’яти пунктів:

а) вимоги. На цьому етапі визначаються подальші вимоги до програмного продукту та його кінцевий вид та призначення;

б) планування. На цьому етапі проводиться планування розробки проекту та шляхи, за допомогою яких можливо буде досягти результат. Визначаються строки, бюджет та оцінюються риски розробки та виявляються зацікавлені сторони;

в) реалізація. На цьому етапі проходить сама реалізацію, тобто розробка програмного продукту. По раніше узгодженому плану проводиться розробка та проводиться контроль по вибраним критеріям;

г) верифікація. На цьому етапі йде тестування готового програмного продукту, та виправлення знайдених помилок. Узгодження виконаної роботи;

д) супроводження. На цьому етапі розроблений програмний продукт передається замовнику.

Цей метод більш за все підходить то проектів, котрі мають чіткі вимоги, або для невеликих та не складних проектів.

1.1.2Agile

Оскільки не для усіх проектів підходить однаковий підхід – існує ще одна система управління [8]. Agile– це гнучка система яка дозволяє розбити проект не на послідовні фази, а на невеличкі під проекти, котрі потім збираються у один проект. Схема Agileметодології представлена на рис 1.2.

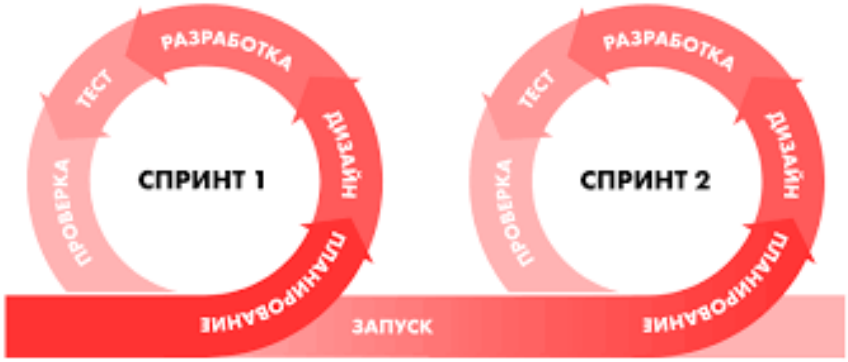


Рис 1.2 –ЖЦ Agile

Сам по собі Agile – не є методом управління проектом. Це скоріш набір ідей та принципів того, як треба реалізовувати проект. На основі Agile методології було створено Scrum, Kanban та інші методології.

Більш за все підходить до проектів, котрі не мають чіткого плану, оскільки є дуже гнучкими.

1.1.3 Scrum

Також, як і Agile, Scrum розділяє проект на невеликі частини, котрі одразу можуть бути використані. Після цього, частини продукту приоритезуються Замовником та передається команді для розробки. Найважливіші частини передаються у першу чергу в розробку в Спринті (рис 1.3) – таку назву має ітерація в Scrumб котра триває 2-4 неділі. У кінці спринту замовнику передається робоча частина програмного продукту. Наприклад сайт з частиною функціоналу, або програма, котра працює, але не повністю.

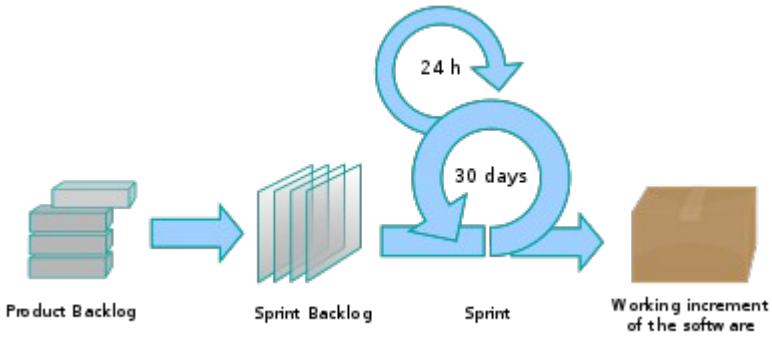


Рис 1.3 – Модель Scrum

Після передачі першої частини продукту замовнику – команда приступає до наступного спрінта. Тривалість спрінта фіксована, на проте, команда обирає її самостійно на початку роботи над простому, сходячи з специфіки проекту на власної продуктивності.

Щоб переконатися у тому, що проект відповідає усім вимогам замовника, котрі можуть змінитися під час розробки продукту, перед кожним спринтом проводиться переоцінка ще не виконаного змісту проекту та внесок у нього змін. У цьому процесі приймає участь уся команда проекту.

Більш за все підходить кваліфікованим та досвідченим командам, котрі вміють розставляти свої пріоритети, та мають чітке представлення о вимогах проекту.

1.2Аналіз предметної області

У сучасному світі, коли існує багата кількість компаній, котрим потрібна система відстеження, помилок ,та їх облік, дуже актуальною становляться багтрекінгові системи. Багтрекінгові системи – програми, котрі розроблені з метою допомогти розробникам програмного забезпечення враховувати та контролювати помилки та недоліки, знайдені у програмних продуктах, бажання користувачів, а також відстежувати процес усунення цих помилок та виконанням або невиконання побажань.

Початок нового проекту, як правило, супроводжується рішенням маси організаційних питань: як будуть взаємодіяти учасники проекту, де будуть зберігатися документи і як буде побудовано їх узгодження, як будуть ставити завдання та видавати доручення.

У кожної компанії, у кожного керівника проектів, вже є заготовки і переваги. Але завжди корисно подивитися, як це роблять інші.

Для організації роботи проектної команди необхідний єдиний інформаційний центр, за допомогою якого вирішуються такі завдання:

- зберігати проектні документи;
- вести робочі матеріали: протоколи, ризики, відкриті питання;
- інформувати учасників про правила, події, плани;
- вести всілякі реєстри - задач, бізнес-процесів, розробок (Excel - не найкращий інструмент для колективної роботи);
- роздавати завдання і доручення;
- збирати інформацію щодо виконання завдань і доручень.

Тож вибір багтрекінгової системи є важливим етапом в розробці ПЗ.

Так, можливо використовувати будь який інструмент, навіть писати на папері (рис 1.1) або писати на електрону пошту.

Та проте, чим більший становиться проект, тим складніше відстежувати усі зміни.

Рис 1.1 – Самий простий баг репорт

Головним критерієм багтрекінгової системи є: CRUD - create, read, update, delete [10]. Тобто, у користувача повинна бути можливість створювати «тікет»

про помилку, можливість його переглядати і в разі потреби редагувати та доповнювати. Так само необхідна можливість видаляти їх, в разі, якщо «тікет» був створю без необхідності.

1.2.1 Ролі

Ролі користувачів визначаються гнучкою моделлю визначення прав доступу користувачів. Ролі включають в себе набір привілеїв, що дозволяють розмежовувати доступ до різних функцій системи.

Користувачам призначається роль в кожному проекті, в якому він бере участь, наприклад, «менеджер в проекті з розробки сайту А», «розробник в проекті з підтримки Інтранету компанії» або «клієнт в проекті по рефакторингу інформаційної системи компанії Б».

Користувач може мати кілька ролей. Призначення ролі для окремого завдання (issue) в даний момент неможливо

1.2.2 Проекти

Проект є одним з основних понять в предметній області систем управління проектами. Завдяки цій сутності можливо організувати спільну роботу і планування декількох проектів одночасно з розмежуванням доступу різним користувачам. Проекти допускають ієрархічну вкладеність.

1.2.3 Трекери

Треки є основною класифікацією, за якою упорядковано завдання в проекті. Саме по собі поняття «трекер» сходить до систем обліку помилок (англ. Bug tracking tool), який представляв кожна окремо один проект.

По суті, треки представляють собою аналог підкласів класу «Завдання» та є основою для поліморфізму різного роду завдань, дозволяючи визначати для кожного їх типу різні поля. Прикладами трекерів є «Поліпшення», «Помилка», «Документування», «Підтримка».

1.2.4 Завдання

Завдання є центральним поняттям всієї системи, що описує якусь задачу, яку необхідно виконати. У кожного завдання в обов'язковому порядку є опис і автор, в обов'язковому порядку завдання прив'язана до трекера.

Кожне завдання має статус. Статуси представляють собою окрему сутність з можливістю визначення прав на призначення статусу для різних ролей (наприклад, статус «відхилений» може привласнити тільки менеджер) або визначення актуальності завдання (наприклад, «відкритий», «призначений» - актуальні, а «закритий», «відхилений» – немає).

Для кожного проекту окремо визначаються набір етапів розробки і набір категорій завдань. Серед інших полів цікаві також «оцінене час», що служить основою для побудови управлінських діаграм, а також поле вибору спостерігачів за завданням.

До завдань є можливість прикріплювати файли (є окрема сутність «Додаток»).

Значення інших перелічуваних властивостей (наприклад, пріоритетність) зберігаються в окремій загальній таблиці.

1.2.5 Відстеження зміни параметрів задач

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 завдання і/або додавання коментаря до неї. Тобто служить одночасно інструментом ведення історії завдання і інструментом ведення діалогу.

Сутність «Змінений параметр» прив'язана до окремого запису журналу і призначена для зберігання старого і нового значення зміненого користувачем параметра.

1.2.6 Зв'язки між завданнями

Завдання можуть бути пов'язані між собою: наприклад, одна задача є під задачею для іншої або передувати їй. Ця інформація може бути корисна в ході планування розробки програми, за її зберігання у програмних продуктах відповідає окрема сутність.

Облік витраченого на проект 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 проект або для оцінки фактичної трудомісткості і вартості розробки.

1.2.7 Термін Bug

Програмна помилка (баг) – означає помилку в програмі або в системі, через яку програма видає несподівану поведінку і, як наслідок, результат. Більшість програмних помилок виникають через помилки, допущені розробниками програми в їх вихідному коді, або в її дизайні. Також деякі помилки виникають через некоректну роботу інструментів розробника, наприклад через компілятора, який виробляє некоректний код. Програму, яка містить велику кількість помилок, серйозно обмежують її працездатність, називають нестабільну

Термін «програмна помилка» зазвичай вживається для позначення помилок, які проявляють себе на стадії роботи програми, на відміну, наприклад, від помилок проектування або синтаксичних помилок.

Звіт, що містить інформацію про помилку також називають звітом про проблему (англ. Bug report). Звіт про критичну проблему (англ. Crash), що викликає аварійне завершення програми, називають краш-репортом (англ. Crash report).

Склад інформації про дефекті:

а) номер (ідентифікатор) дефекту;

б) короткий опис дефекту;

в) хто повідомив про дефект;

г) дата і час, коли був виявлений дефект;

д) версія продукту, в якій виявлений дефект;

е) серйозність (критичність) дефекту і пріоритет рішення;

є) опис кроків для виявлення дефекту (відтворення ненавмисного поведінки програми);

ж) очікуваний результат і фактичний результат;

з) хто відповідальний за усунення дефекту;

і) обговорення можливих рішень і їх наслідків;

й) поточний стан (статус) дефекту;

к) версія продукту, в якій дефект виправлений.

Як правило, система відстеження помилок використовує той чи інший варіант «життєвого циклу» помилки [5], стадія якого визначається поточним станом, або статусом, в якому знаходиться помилка.

Типовий життєвий цикл дефекту:

а) новий - дефект зареєстрований тестувальником;

б) призначений - призначено відповідального за виправлення дефекту;

в) дозволений - дефект переходить назад в сферу відповідальності тестувальника. Як правило, супроводжується резолюцією, наприклад:

1) виправлено (виправлення включені в версію таку-то);

2) дубль (повторює дефект, вже знаходиться в роботі);

3) не виправлено (працює у відповідності зі специфікацією, має занадто низький пріоритет, виправлення відкладено до наступної версії і т. п.);

4) не відтворюваність (запит додаткової інформації про умови, в яких дефект проявляється).

г) далі тестувальник проводить перевірку виправлення, в залежності від чого дефект або знову переходить в статус призначений (якщо він описаний як виправлений, але не виправлений), або в статус закритий;

д) відкритий повторно - дефект знову знайдений в іншій версії.

Та проте, життєвий цикл може змінюватися в залежності від особистостей проекту.

1.3Загальні правила роботи з багтрекінговими системами

У ході тестування проекту може виникнути велика кількість помилок, та щоб розробник не пропустив ваші зауваження, необхідно дотримуватися деяких правил:

а) обрати тип:

1) баг – помилка у роботі програми;

2) покращення – програма працює коректно, та проте, її можливо покращити\доповнити;

3) нова розробка.

Наприклад, замовник бажає додати новий функціонал, та у системі це було визначено як баг, и розробники розробляли інші частини нового функціоналу, та не дійшли до цього. Як результат – не виконана важлива робота. Цього можливо було уникнути лише обрав правильний тип. Наприклад таблиця 1.1:

Таблиця 1.1 – класифікація типів

Баг	Покращення	Нова розробка
Помилка 500 при завантаженні xlxs файлу	Виводити усі допустимі типи файлів	Можливість завантажувати декілька файлів одночасно
Город Київ виправляється на Києв	Виводити індекс у результатах вибору адреси	Обробка іноземних адрес
При завантаженні файлу великого розміру повідомлення про помилку «некоректний тип файлу»	Збільшити обмеження розміру файлу до 30Мб	Загрузка файлів формату .mp4

б) треба давати короткий але інформативний заголовок;

Коли розробник, або менеджер проекту проводе аналітику заведених помилок, то він повинен мати можливість по назві зрозуміти, що це за помилка, та наскільки вона критична;

в) при наявності можливості, треба додати скріншот.

Скріншот може допомогти як розробнику знайти баг та виправити його, тому як виправлення цієї помилки може початися через місяць, коли інтерфейс змінився, так и тестувальнику, щоб згадати, яка була помилка;

г) дуже важливо детально описати кроки до відтворення результату.

Якщо для розробника не достатньо скріншоту – він буде читати опис кроків для відтворення. Кроки повинні бути короткі, інформативні та послідовні;

д) описати очікуваний результат. Оскільки тестувальник вважає, що це баг – він повинний описати очікуваний результат. Важливо описати, чому саме це помилка? Згідно документації або в іншому місці це працює інакше.

Усі ці 5 пунктів є обов’язковими для кожного баг-репорту, [15] саме тому, кожна багтрекінгова система має надавати можливість реалізувати усі ці пункти. Далі будуть розглянути багтрекінгові системи, які надають можливість реалізації кожного з цих пунктів.

2ПОСТАНОВКА ЗАВДАННЯ ДОСЛІДЖЕННЯ

2.1Дефекти у додатку

Коли о багах повідомляють, це може складатися з багатьох завдань, таких як:

- розслідування;
- збір інформації;
- тестування;
- виправлення помилок.

Усе це може проходити упродовж усього циклу розробки проекту. Дуже не зручно для управління питаннями проекту, якщо для кожного знайденого багу буде заведено окремий документ, оскільки за період розробки проекту може бути знайдено сотні помилок.

2.2 Комунікація між користувачами

QA команда, та команда розробки потребують більш ефективної комунікації. Дослідження [3] виявило, що намагатися тримати у пам’яті інформацію про усі помилки, або у одному файлі, може викликати лише катастрофу. Тому, що ви не маєте можливості спілкуватися ефективно з усією командою розробки. Також, у міру зростання з’являється проблема, що тільки одна людина має можливість змінювати електрону таблицю[4]. Не залежно від того, наскільки технічно компетентними є члени команди, якщо комунікація між командою тестувальників та розробників не ефективна – уся команда може потерпіти невдачу

2.3 Складність

Для цього існує багато схожих систем відстеження помилок. Проте де яки з цих систем мають складний та заплутаний функціонал. Більша частина відстеження помилок побудована у вигляді веб-додатків. У веб-додатках може бути важко орієнтуватися, особливо якщо вони побудовані як послідовність HTML – сторінок, ніж як додаток. Деякі веб-додатки перезавантажують сторінку загалом після введення даних, оскільки призначені для зв’язку між користувачем та веб-сервісом. Це призводить до того, що робота сповільнюється.

2.4 Постановка задачі

Завдання, яке ставить перед собою дана робота – проаналізувати та порівняти існуючи алгоритми багтрекінгових систем. Для цього буде проведено аналіз існуючих алгоритмів багртекінгових систем. Спочатку буде проведено дослідження та аналіз предметної області, для загального розуміння тематики, у котрій буде проведено досліди. Далі буде розглянуто існуючи багтрекінгові системи, та проведено їх порівняння за певними критеріями. Також буде проведено аналіз та порівняння існуючих методів управління програмними продуктами.

На підставі отриманих результатів, буде алгоритм визначення придатності тієї чи іншої багтрекінговою системи для її використання у конкретному проекті. На основі котрого, будь-яка компанія або команда зможе обрати багтрекінгову систему, котра найліпше підійде для кожного окремого проекту.

3 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ

3.1 Аналіз багтрекінгової системи Bugzilla

Інструмент Bugzilla був розроблений, коли проекту веб-браузера з відкритим вихідним кодом Mozillабув потрібен інструмент для відстеження помилок. Він використовувався в якості внутрішнього інструменту для відстеження помилок в Mozilla.

Інструмент спочатку був закодований з використанням мови програмування Perl, який пізніше пройшов ряд поліпшень спільнотою програмістів.

Bugzilla з'явився як надзвичайно популярний інструмент, який використовується Apache, Open Office, Linux, а також приватними організаціями, які розробляють критично важливі системи, такими як NASA і IBM. Bugzilla дозволяє користувачам додавати нові знайдені помилки, редагувати і відстежувати існуючі.

Поля Product, Component, Version, Status і Report дозволяють користувачам відстежувати помилки. Навпаки, Summary, Whiteboard, Severity, Attachment sand the Dependency дозволяють користувачеві виправляти помилки.

Інструмент також дозволяє користувачам шукати інформацію, використовуючи фільтри, а також через звіти Bugzilla, в яких представлена докладна інформація, пов'язана з статусом помилок і їх вплив на додаток через графічне представлення.

Цей веб-додаток підтримує MySQL і HTML сторінки які використовуються для взаємодії з користувачем.

Bugzilla в даний час є однією з найпопулярніших систем для відстеження дефектів, що охоплюють більшість функцій, очікуваних від ефективної системи стеження за вадами.

Основні функціональні можливості інструменту включають в себе:

- основні і розширені можливості пошуку. Удосконалена система пошуку дозволяє налаштовувати пошук по часу, пріоритету і різним іншим характеристикам;

- Email повідомлення допомагає відслідковувати будь-які зміни, зроблені в Bugzilla;
- система автоматичного виявлення дублікатів помилок, яка шукає схожі помилки в системі.

Система шукає існуючі помилки з ідентичною назвою. Результати пошуку можуть бути збережені і перезапущені за допомогою натискання кнопки опції;

- можливість створювати і змінювати існуючу помилку по електронній пошті;
- система відстеження часу усунення помилки;
- можливість переглядати всіх користувачів Bugzilla, «призначати» помилки, запитувати в користувачів звіт, результати роботи над помилкою, а також спостерігати, допомагати іншим розробникам, які перебувають у відпустці або не можуть приступити до усунення помилки з інших причин;
- можливість встановлювати помилки однієї установки Bugzilla в іншу, а також в декількох версіях інструмент.

Основні переваги Bugzilla:

- ефективне відстеження помилок і змін в коді;
- огляд нових розроблених або оновлених патчів коду;
- можливість скоротити час простою;
- допомагає в підвищенні продуктивності;
- збільшує задоволеність клієнтів;
- забезпечує підзвітність;
- покращує спілкування;
- знижує вартість і підвищує якість продукції.

ЖЦ кожного тікету в Bugzilla досить варіативний (див рис 1.2), тому дозволяє детально відстежувати етап, на якому знаходиться розробка кожного компоненту. Завдяки такій гнучкій системі, можливо відстежувати загальну кількість репортів на кожному проекті, та їх статуси.

Рис 1.2 – ЖЦ Bugzilla

Bugzilla – це інструмент, який постійно і послідовно розробляється і тестується для подальшого і постійного поліпшення. Завдяки цій стратегії розвитку, інструмент був прийнятий провідними технологіями в світі.

Bugzilla підійде для великих проектів, оскільки має багатий функціонал. Та для не великих проектів може бути зайвим більша частина функціоналу, та відволікати від дійсно важливих речей.

3.2 Аналіз багтрекінгової системи REDMINE

Redmine – відкрите серверне веб-додаток для управління проектами та завданнями (в тому числі відстеження помилок).

Redmine написаний на Ruby і являє собою додаток на основі широко відомого веб-фреймворку Ruby on Rails.

Даний продукт містить наступні можливості:

а) ведення декількох проектів;

б) гнучка система доступу, заснована на ролях;

в) система стеження за вадами;

г) діаграми Ганта (рис 1.3) і календар;

д) ведення новин проекту, документів і управління файлами;

е) оповіщення про зміни за допомогою RSS-потоків і електронної пошти;

є) форуми для кожного проекту;

ж) облік тимчасових витрат;

з) настроюються довільні поля для інцидентів, тимчасових витрат, проектів і користувачів;

и) легка інтеграція з системами управління версіями (SVN, CVS, Git т.д.);

і) створення записів про помилки на основі отриманих листів;

ї) підтримка множинної аутентифікації LDAP;

й) можливість самостійної реєстрації новий користувачів;

к) багатомовний інтерфейс;

л) підтримка СУБД (MySQL, Microsoft SQL Server, PostgreSQL, SQLite).

Рис1.3–Діаграма Ганта

Redmine надає можливість інтеграції з різними системами управління версіями (репозиторіями). Інтеграція полягає у відстеженні змін у зовнішньому репозиторії, їх фіксації в базі даних, аналізі змін з метою їх прив'язки до певних завдань.

У інфологічній структурі системи за інтеграцію з зовнішніми репозиторіями відповідають три сутності: репозиторій, редакція і зміна.

а) сховище - пов'язана з проектом сутність, що зберігає тип підключеного сховища, його місцезнаходження та ідентифікаційні дані його користувача;

б) редакція - відображення редакції сховища, і, крім інформаційних полів, може бути прив'язана до конкретного завдання: для цього потрібно вказати в описі змін «refs #NUM», де NUM - номер завдання), і до користувача-автору редакції;

в) зміна - зберігає список змінених (доданих, віддалених, переміщених, модифікованих) файлів в кожній редакції.

Також система має повідомлення користувачів про зміни, що відбуваються на сайті, здійснюється за допомогою сутності «Спостерігачі», що зв'язує користувачів з об'єктами різних класів (проекти, 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**.**

До недоліків Redmine можно віднести:

а) управління файлами і документами в Redmine зводиться до їх додаванню, видалення та редагування. Правами доступу ні до файлів, ні до окремих документів управляти не можна;

б) у Redmine можна управляти правами доступу на рівні окремих полів завдання. Наприклад, на даний момент від клієнтів не можна приховати оцінки часу роботи над завданням. Але можна зробити додаткові поля видимими тільки користувачам з певними ролями;

в) у Redmine в список завдань не виводиться загальна трудомісткість задач;

г) немає можливості дати користувачеві роль у всій системі; наприклад, «Керівник проектного офісу» повинен мати доступ до всіх проектів в системі: для цього потрібно додати користувача з цією роллю в усі проекти;

д) підключити git репозиторій можливо тільки в разі, якщо і Redmine, і репозиторій знаходяться на одному сервері.

Redmine– система, котра підійде більш для невеликих проектів, або до проектів з чіткими вимогами, тобто проекти, що розробляються за каскадною моделлю.

3.3 Аналіз багтрекінгової системи MANTIS

Mantis – це безкоштовне, доступне для скачування програмне забезпечення для відстеження помилок з відкритим вихідним кодом. MANTIS був розроблений в кінці 2000 і може бути використаний різними проектами розробки програмного забезпечення. Забезпечує взаємодію розробників з користувачами (тестувальниками). Дозволяє користувачам заводити повідомлення про помилки і відстежувати подальший процес роботи над ними з боку розробників.

Система має гнучкі можливості конфігурації (рис 1.4), що дозволяє налаштовувати її не тільки для роботи над програмними продуктами, а й в якості системи обліку заявок для Helpdesk.

Рис 1.4 – розширена конфігурація Mantis

Можлива інтеграція з wiki-движком для створення документації (DokuWiki).

Назва Mantis (богомол) походить від того, що богомол харчується жуками (bug).

До плюсів Mantis:

- безоплатність і вільність, ліцензія GNU General Public License (GPL);
- код на PHP вільно модифікуємо;
- зрозуміло написаний код;
- колірна індикація по статусу інциденту (бага);
- налаштовані користувачем поля;
- зручні фільтри;
- швидкість роботи;
- повідомлення по e-mail;
- велика кількість плагінів, що розширюють функціональність;
- можливість відправити нагадування (рис 1.5)

Рис 1.5 Відправлення нагадування

У той же час, до мінусом даного програмного продукту можна віднести:

- через веб-інтерфейс можна зробити суттєві зміни налаштувань. Необхідно налаштовувати в конфігурації;
- через інтерфейс можна редагувати можливість переходу між статусами, але не список статусів;
- змінити (додати, видалити) наявні поля в фільтрі, вікнах створення і перегляду бага можна тільки редагуючи код. Але дані операції з кодом досить прості і не вимагають глибоких знань програмування на PHP.

Система є веб-додатком, тому не вимагає для роботи спеціального ПЗ на стороні клієнта і працює через веб-браузер. Ця система є зручно

3.4 Аналіз багтрекінгової системи Trac

Trac – це проект з відкритим вихідним кодом, який використовується для відстеження проблем в проектах розробки програмного забезпечення. Система забезпечує взаємодію з різними системами контролю версій, такими як Subversion і Git, допомагаючи поліпшити можливості звітності. Trac має вбудований потужний движок візуалізації вики і допомагає в наданні опису проблем. Інструмент допомагає в наданні посилань і посилання між помилками, завданнями, наборами змін, файлами і вікі-сторінками. Основні функції інструменту включають в себе:

- працює як легкий проект управління і простий у використанні;
- інструменти мають гнучкість з різними типами проектів і можуть бути налаштовані за допомогою плагінів або написаним користувачем кодом;
- чи відстежує дозвіл помилок, проблем, запитів функцій за допомогою системи тікетів;
- можливість призначати ступінь серйозності тікетів, а потім фільтрувати їх;
- моніторинг змін у проекті шляхом надання графіка;
- підсвічування коду і порівняння файлів;
- доступ до анонімних користувачів для перегляду статусу проектів, заявок і етапів, хоча потрібна реєстрація для внесення змін через «систему дозволів»;
- має Документ-сервер, який допомагає в спілкуванні, управлінні користувачами і ресурсами;
- має настроюється шаблон і розширюваний плагін, який допомагає користувачам налаштовувати інструмент відповідно до потреби.

Функція тимчасової шкали відображає всі поточні і події попередніх проектів полегшували відстеження дій проекту. Використовується на проектах, де можуть бути часті зміни.

3.5Аналіз багтрекінгової системи JIRA

Є нескінченне число варіантів рішення - разом чи окремо. Можливо використовувати зв'язку Confluence + JIRA, це може бути зручно та ефективно.

Маса корисної інформації можливо організувати у вигляді спеціальних сторінок. Наприклад, для великої команди вкрай популярною є контактна карта, на якій можливо зробити список проектної команди по групах, з фотографіями і даними учасників.

3.5.1Портал проекту – Confluence

Confluence - це зручний і просунутий wiki движок від компанії Atlassian. Він дозволяє організувати внутрішній Інтернет портал і дати доступ до нього всім користувачам - для редагування або для читання:

- він дуже простий і зручний, для навчання досить декількох годин. У нас на проекті сторінки створювали і редагували майже всі учасники;
- досить багаті можливості форматування, необхідні для того щоб зробити сторінку красивою і легко читається. Є кошти, що автоматизують створення навігації усередині сайту - змісту, таблиці, посилання, включення уривків з інших сторінок і т.д;
- написано величезна кількість плагінів для розширення функціональності;
- можна зберігати документи, при цьому зберігаються версії. За замовчуванням користувач бере завжди останню версію, що знижує кількість помилок. У будь-який момент можливо повернутися до будь-якої з попередніх версій;

- також зберігаються версії сторінок, і завжди можна побачити, хто і які зміни вніс, порівнюючи будь-які дві версії попарно;
- можна обмежувати доступ в цілому на сайт проекту або на окрему сторінку;
- повнотекстовий пошук здійснюється по всіх сторінках і вкладеним документам порталу, включаючи pdf.

Наприклад, можливо зробити Confluence проектний портал, домашня сторінка якого містить посилання на ключові матеріали проекту, контактну карту, регламенти та інструкції. На цій же сторінці публікуватимуся всі новини проекту.

3.5.2 Jira

Сторінка з посиланнями на екземпляри системи підтримувалися системними адміністраторами. Там може бути схема еволюції технічної інфраструктури - коли якісь екземпляри з'являлися і виводилися з експлуатації.

Величезна кількість компаній і команд вважають за краще використовувати в якості багтрекінгову систему Atlassian JIRA.

Головною перевагою jira є можливість її повністю налаштувати під свої потреби: Workflow (рис 1.6), фільтри, форми, стану.

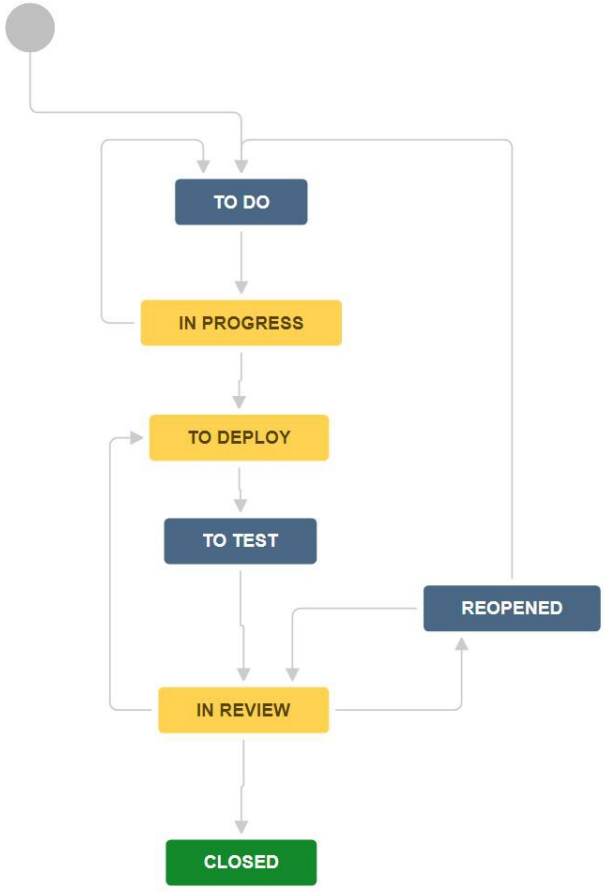


Рис.1.6–Приклад простого Workflow

Як би не було парадоксально, але це є одночасно і великим недоліком системи. Психологія людей така – якщо є налаштування, значить, їх обов'язково потрібно налаштовувати. І якщо це робити не професійно, то в подальшому при роботі над проектами можуть виникати проблеми:

- ускладнюються Workflow завдання і форми, що веде до ускладнення роботи з самою системою; (Рис. 1.7)

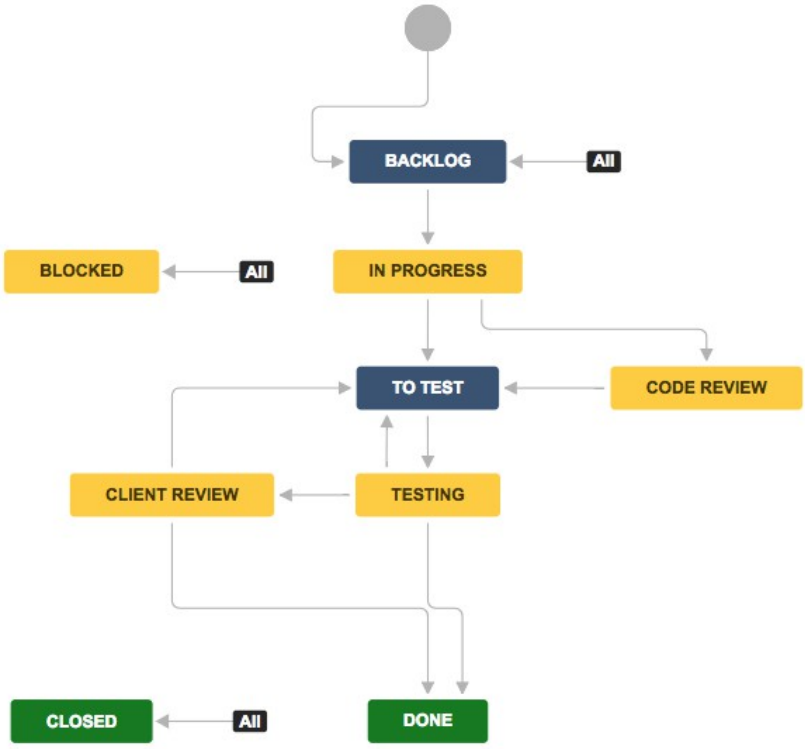


Рис1.7–модіфіковане Workflow

- через багатофункціональності іноді важко знайти те, що необхідно;
- плагіни доступні тільки у версії для завантаження, та в hostedверсії відсутня інтеграція з github та Google calendar;
- складна у освоєнні;
- наприклад, створені нові поля форми, в подальшому не використовуються, незабаром перетворюються в непотрібне сміття, який забиває і так перевантажений інтерфейс;

Але позитивних моментів, все ж, набагато більше:

- абсолютно точно JIRA є лідером в області продуктів менеджменту і багтрекінга, ідеально підходить розробникам програмного забезпечення. Є можливість налаштування для будь-якої сфери діяльності проекту або всієї компанії (якщо вміло цим користуватися)(рис 1.8):

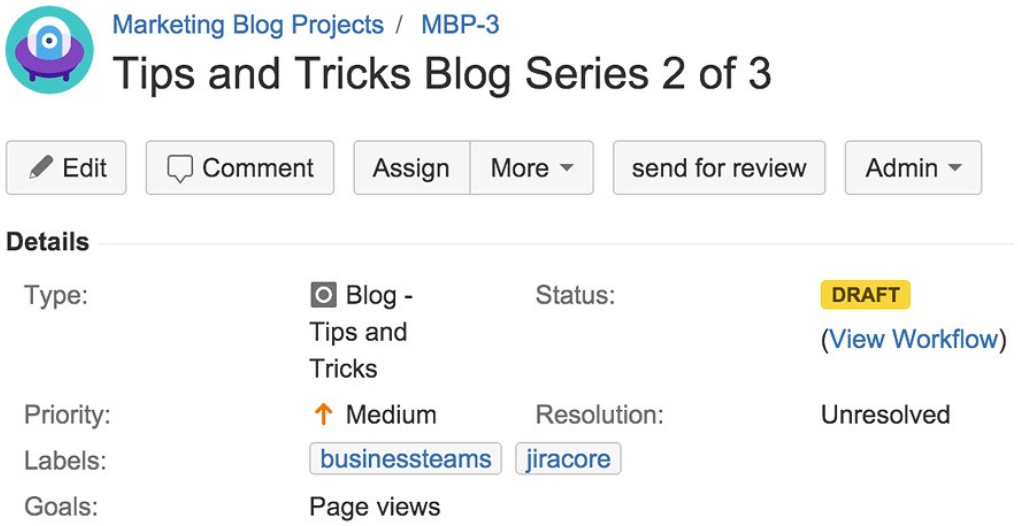
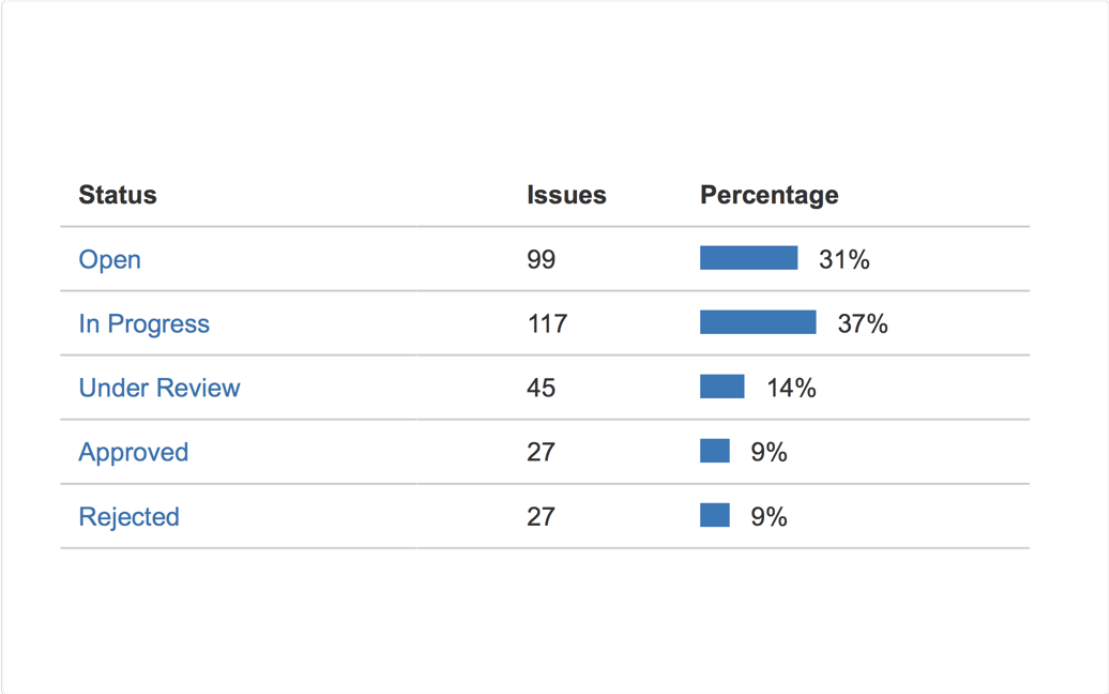


Рис1.8–Деталізація задач

- великою перевагою JIRA над open-source додатками, є наявність API і безлічі плагінів, з цього можливості розширень не обмежені. Існує більше 100 готових безкоштовних розширень, не кажучи про те, що завжди є можливість написання власних;
- незамінний інструмент для побудови будь-яких звітів і аналітики за завданнями і проектами;
- відмінна система для управління завданнями в багатомовному сегменті;
- хороша система постановки, а так само виконання завдань. Вона дозволяє складати план роботи на день, тиждень, місяць, а так само планувати свій час на вирішення робочих завдань. Підходить для використання не тільки в роботі, але і в особистих цілях;
- є можливість відстежувати проблеми проекту і стежити за ходом виконання кожного завдання, а також контролювати виконання поставлених завдань будь-яким виконавцем в будь-який проміжок часу Рис (1.9);



Status	Issues	Percentage
Open	99	31%
In Progress	117	37%
Under Review	45	14%
Approved	27	9%
Rejected	27	9%

Рис1.9–Статус тікетів

- важливий фактор, це високий рівень безпеки;
- ця система підходить для організацій як з малою кількістю співробітників (до 10 осіб), так і для більших компаній (до 200 осіб);
- систему можна встановити на смартфон;
- присутній зручний і варіативний пошук (рис 1.10);

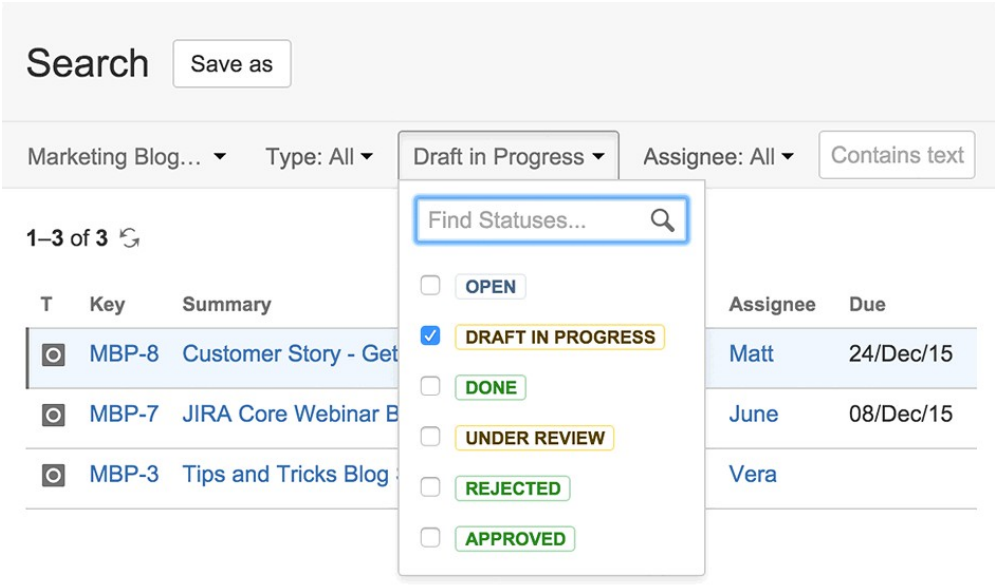


Рис1.10–Фільтр пошука

– зручна система повідомлень. Можна використовувати @ згадки для залучення уваги конкретних учасників команди і залишайтеся в курсі справ, отримуючи зручні і докладні повідомлення. І користувач відразу отримує повідомлення про нове завдання \ повідомленні \ оновленні (рис 1.11.);

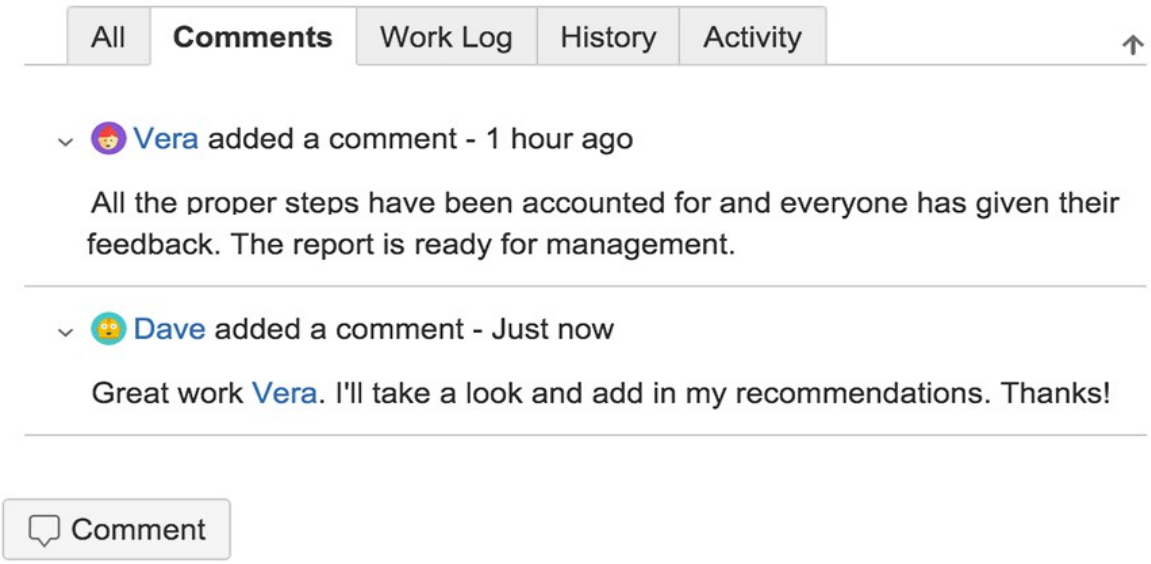


Рис1.11–Відстеження коментарів

– також важливо, що при створені багу\завдання, є велика кількість налаштувань (рис 1.12), що допомагає у пошуку та приоритезації задач.

Рис 1.12 – поля створення Bug

Ну і наостанок ще один мінус це ціна. Важливу роль у виборі системи грає вартість програми. Складно змагатися з безкоштовними продуктами, але порівнюючи всі перераховані вище плюси і мінуси системи, можливо сказати одне -Atlassian JIRA варто того. Також

3.6 Аналіз багтрекінгової системи GanttPRO

GanttProject – це програмний продукт, який призначений для планування проектів на основі будування діаграми Ганта та діаграми типу PERT. Підтримується імпорт та експорт документів Microsoft Project.

Завдяки зручному інтерфейсу (рис 1.13), можливо відстежувати ресурси команди, та зайнятість кожного чоловіка в команді.

Рис 1.13 – Інтерфейс проектів

До переваг можна віднести:

- зручний, та інтуїтивно зрозумілий інтерфейс;
- співвідношення ціна\якість;
- інтерфейс та служба підтримки на різних язиках;
- підходить для управління персональними та командними проектами;
- можливість переключитися на дошку задач;
- інтеграція з Jira Cloud;
- готові шаблони для швидкого початку роботи.

Проте, до недоліків можливо віднести:

- відсутність подивитися усі проекти на одному екрані;
- недостатньо інтеграцій.

Це доступний, та зручний інструмент для роботи у невеликих командах, та зручний для відстежування зайнятості кожного окремого працівника.

3.7 Аналіз системи Trello

Попрете, що Trello не є багтрекінговою системою її теж варто розглянути, оскільки невеликі компанії та команди використовують цей програмний продукт, для відстежування етапів проекту. Завдяки своїй простоті у використанні, та

інтуїтивно зрозумілому інтерфейсу (див рис 1.14), Trello використовують коли кількість працівників не велика, та у кожного своя чітка роль на проекті.

Рис 1.14 – Інтерфейс Trello

До переваг Trello можливо віднести:

- дуже легко розібратися;
- безкоштовної версії достатньо для більшості проектів;
- швидкий та зручний пошук;
- метки для створення категорій;
- для досвідченого користувача є покращення;
- інтерфейс та підтримка на декількох язиках.

Та проте, до мінусів можно віднести:

- не ефективність на довгострокових проектах;
- акцент на поточній роботі, а не на датах, пріоритетах або дедлайнах.

Не дивлячись на велику кількість переваг, Trello являється більш допоміжним програмним продуктом для окремих команд.

4 КЛАСИФІКАЦІЯ ТА ПОРІВНЯННЯ АЛГОРИТМІВ СИСТЕМ

4.1 Цільові алгоритми

Коли розробник або тестувальник кожен день працює к багтрекінговою системою, він бажає бачити задачі, котрі потребують його уваги, та назначені йому, а також ті задачі, котрі потребують найшвидшого реагування [12]. Якщо буде велика кількість інформації, котра може торкатися проекту загалом, та не окремого розробника або тестувальника.

Система повинна виводити на передній план лише не розрішені задачі, а не усі загалом. Загальні задачі, котрі не пов’язані з помилками, можуть відображатися, та в окремому місці.

Для менеджера проекту, або тестувальника є важливим комунікація за допомогою системи. Такий дуже зручною частиною системи є те, що користувач бачить строк, до котрого він повинен виконати ту чи іншу роботу.

Як можливо побачити, у наш час, багтрекінгова система, це більше, ніж просто система, яка відслідковує нові помилки, а ще й виконує роль комунікації між командою, та пам’яткою о справах, котрі необхідно зробити.

На проекті може існувати багато ролей, та кожний буде працювати з системою по своєму. Тож багтрекінгова система представляє різні речі для різних людей. Ці маленькі, та проте значимі відмінності по виявлятися в системі через функції, задовольняючи кожну з індивідуальних відмінностей кожного з зацікавлених сторін.

Налаштовані рольові інтерфейси, котрі підкреслюють певні аспекти трекера при абстрагуванні від інших, можуть краще відповідати безлічі зацікавлених сторін, котрі складають аудиторію системи відстеження помилок[13].

Кожна знайдена помилка у системі має зайняти своє місце у багтрекінговій системі. Котра буде працювати звітом о її існуванні, та можливістю її вирішити.

Існують різноманітні ступені існування помилки від першого повідомлення про помилку до підтвердженого, та до вирішеного статусу помилки.

Представлення полегшених можливостей тегірування для відстеження проблем, може полегшити вибір між великою кількістю детальних та необхідних

полів, зв’язаних з проблемами та підтримку спрощеного інтерфейсу для створювання справ. Ці теги повинні використовувати існуючу надійну інфраструктуру багтрекінгової системи для пошуку, сортування та фільтрації питань.

У команді розробників програмного забезпечення існує декілька рівнів володіння проблемами. Використовуючи шаблон у самому трекері, зацікавлені сторони можуть позбавитися від їх необхідності вручну відстежувати помилки, котрі ще існують.

4.2Критерії класифікації

Класифікація – це система розподілення об’єктів по раніше визначеними групам ознак.

Для того, щоб визначити, котра з систем є найбільш оптимальною у використанні, треба визначити критерії, по яким буде проводитись порівняння:

а) розширений пошук. Розширений пошук допомагає на великих проектах, та у великих компаніях, де над кожним проектом може працювати велика кількість людей. За допомогою розширеного пошуку, є можливість швидко віднайти потрібний тікет, або потрібний проект. Ця функція є дуже важливою для великих проектах та не є необхідною на невеликих проектах;

б) високий рівень захисту. Високий рівень захисту є необхідним та майже ключовим фактором, коли іде розробка проекту. Проте, якщо продукт розробляється без цілі його продажу, та з відкритим кодом, цей пункт не є основним;

в) трекер часу є зручною функцією, для пошуку необхідної інформації та є важливою частиною, коли замовник додає побажання до продукту,та є час, до якого функціонал має бути реалізовано;

г) Email повідомлення допомагає відстежувати зміни у проекті. Наприклад коли розробник обновив продукт, команда тестувальників (або сам