Титульник.

Реферат.

Задание по ДП

Содержание.

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

ПО – программное обеспечение

ПС – программное средство

ПП – программный продукт

СОДПП – система отслеживания дефектов программных продуктов

ПК – персональный компьютер

ОС – операционная система

ЭТ – эргономические требования

СЧКС – система «человек-компьютер-среда»

БД – база данных

ВВЕДЕНИЕ

В ходе данного дипломного проекта будет разработана информационная система отслеживания дефектов программных продуктов и произведен ее глубинный анализ, что позволит не только расширить, углубить теоретические и практические знания в инженерно-психологическом проектировании, но и усовершенствовать навык принятия правильных решений по разработке программных средств.

Актуальность выбранной темы обуславливается интенсивной автоматизации работ трудно поддающихся формализации и общей автоматизации разработки ПО. В контексте информационных технологий это означает потребность в разработке и усовершенствовании эффективных сервисов, которые направлены на облегчение работы разработки программного обеспечения в целом, и отдела тестирования в частности.

Исходя из динамики прошлых лет видно, что рынок требует не только усовершенствования старых систем, но и разработки совершенно новых. Их качество должно соответствовать довольно высоким требованиям, ведь учет и дальнейшая работа с дефектами ПС является неотъемлемой частью производства; упущенные или необработанные дефекты могут привести к серьезным временным и материальным потерям, в крайнем случае — судебным разбирательствам, что, очевидно, губительно для любого производства или бизнеса.

Разрабатываемое ПО несет цель облегчить и систематизировать часть процессов, связанных с отслеживанием и учетом программных дефектов. Это включает в себя частичную автоматизацию таких задач, как отслеживание и управление дефектами программных продуктов, анализ их воздействия, управление приоритетами, мониторинг процессов устранения дефектов, а также оценка эффективности и вовлечение членов команды разработки. Автоматизируя эти процессы, организации могут уменьшить количество ошибок, связанных с управлением дефектами, повысить точность информации, повысить эффективность и, в конечном итоге, улучшить общий опыт участников в разработке программного обеспечения.

Цель разработки модели данного программного средства заключается в предоставлении базового решения, что в дальнейшем позволит разработать, и удовлетворить обще-рыночную потребность. Данное ПС позволит структурированно хранить, обрабатывать и изменять данные необходимые для функционирования отдела тестирования предприятия или другого отдела, схожего по назначению.

1. ИНФОРМАЦИОННЫЕ СИСТЕМЫ УЧЕТА И ОТСЛЕЖИВАНИЯ ДЕФЕКТОВ ПРОГРАММНЫХ ПРОДУКТОВ

1.1 Системы учета и отслеживания дефектов программных продуктов в процессе разработки программного обеспечения. Анализ предметной области

Область разработки ПО постоянно расширяется и вместе с ней растет процент программ, используемых в самом процессе цикла разработки.

На сегодняшний день компании, занимающиеся разработкой программного обеспечения, сталкиваются с растущим объемом кода, частыми обновлениями и необходимостью поддерживать высокий уровень стабильности продукта. Тестирование, исправление дефектов и интеграция нового функционала являются неотъемлемой частью работы любого разработчика и беспрерывно сопровождают продукт на каждой стадии цикла разработки программного обеспечения.

На этапе тестирования современного цикла разработки программного обеспечения могут возникнуть различные проблемы в контексте учета и отслеживания дефектов:

* замедление процесса исправления ошибок, назначения персонала и контроля состояния дефектов, связанные с отсутствием совместной среды учета дефектов;
* затруднение взаимопонимания и рабочих коммуникаций в связи с отсутствием согласованного формата и стандартизированного шаблона содержания отчета;
* необходимость дополнительных временных затрат, связанных с отсутствием приоритизации и детального описания дефектов;

Системы отслеживания дефектов решают все из перечисленных проблем, и являются неотъемлемой частью инфраструктуры разработки программных продуктов, предоставляя возможность эффективного управления дефектами от момента их обнаружения до их полного исправления. Однако, большинство существующих программных решений разработаны с расчетом на использование в крупных компаниях, и не поддерживают возможность автономной работы.

Разрабатываемый программный продукт нацелен на упрощение и систематизацию определенных этапов процесса отслеживания и учета дефектов в программном коде. Это включает частичную автоматизацию таких задач, как мониторинг и управление дефектами программных продуктов, анализ их воздействия, управление приоритетами, отслеживание процессов устранения дефектов, а также оценку эффективности и вовлечение участников команды разработки.

За счет интеграции системы отслеживания дефектов в рабочее окружение разработки, организации способны уменьшить количество ошибок, связанных с управлением дефектами, повысить точность предоставляемой информации, улучшить эффективность, повысить общий уровень автоматизации трудового процесса, а так же облегчить участникам цикла разработки некоторые из повседневных задач.

При тестировании программных продуктов, а так же в процессе внедрения нового функционала, разработкой которого занималось более одного человека, очевидна необходимость коммуникации и общего взаимодействия работников. В разрезе разрабатываемого решения коммуникация не обязательно должна выглядеть как групповое обсуждение дефекта [1], а может представлять собой просто опосредованное общение в виде отчета по дефекту или, в нашем случае, экземпляр учтенного дефекта в СОДПП.

Большинство из существующих приложений доступны через веб-интерфейс, что делает их доступными из любого браузера без необходимости установки дополнительного программного обеспечения, но и делает их менее гибкими в замкнутых коммерческих средах, в которых по каким-либо причинам ограничен доступ к внешнему окружению. Для хранения информации о дефектах используются различные типы баз данных, такие как SQL – подобные или NoSQL, что зависит от потребностей масштабируемости и производительности проекта.

Для реализации вышеперечисленного функционала достаточно одной общей базы данных вне зависимости от ее класса [2], она может быть как централизованная, так и распределенная. Современным и наилучшим решением будет применение локальной базы данных на ПК каждого пользователя, которая может быть в дальнейшем связана между собой — это может быть как посредник в виде общего сервера, так и прямая связь между клиентами пользователей. В существующих решениях чаще всего используется первый вариант, а сервер для обработки запросов от пользователей и управления данными в основном [3] используются следующие языки программирования:

* JavaScript и TypeScript;
* Python;
* C# и платформа .NET;
* Java;
* Ruby (Ruby on Rails);

Тем не менее, приложение обычно разрабатывается с уже известным команде разработчиков набором (далее – стеком) технологий, или максимально похожим на него. Это происходит в силу принятой индустриальной практики, при которой команды разработчиков стремятся к максимальной эффективности и оптимизации процесса создания программного обеспечения.

Если стек технологий достаточно гибок для реализации требуемого функционала и способен беспроблемно интегрироваться в требуемое окружение, то предпочтение отдается знакомым инструментам, что позволяет избежать излишних затрат времени и ресурсов на изучение новых стеков для каждого нового проекта. Такой подход содействует ускорению процесса разработки, минимизации рисков и обеспечивает более высокое качество конечного продукта благодаря глубокому знанию выбранного стека технологий со стороны команды разработчиков. Анализируемая сфера разработки программного обеспечения так же подчинена такому подходу.

Поскольку данные о дефектах могут содержать чувствительную информацию о продукте или организации, некоторые из крупных компаний закупают решения для сугубо внутреннего пользования, что существенно затрудняет доступ к некоторой статистике.

Разрабатываемый программный продукт предоставит базовое решения, что в дальнейшем позволит удовлетворить обще-рыночную потребность. Данный ПП позволит структурированно хранить, обрабатывать и изменять данные, необходимые для функционирования отдела тестирования предприятия или другого отдела, схожего по назначению. Преимуществами системы являются ее легкая модифицируемость и масштабируемость.

1.2 Существующие аналоги системы отслеживания дефектов программных продуктов

В области отслеживание дефектов ПП наблюдается некоторая стагнация — большинство представленных в свободном доступе вариантов были разработаны более десяти лет назад. Список популярных существующих программных решений ограничивается списком из четырех названий:

1. Bugzilla
2. JIRA
3. Trac

4. The Bug Genie

Далее рассмотрим их более детально, вкратце описав и выделив ключевые характеристики в каждом из них:

1. Bugzilla – это одна из наиболее знаменитых систем отслеживания ошибок и дефектов в программном обеспечении. В 1998 году Netscape представила первую версию этой системы в общий доступ. Bugzilla является свободным ПО и распространяется по публичной лицензии, то есть пользователи имеют права на его неограниченную установку, запуск, свободное использование [4], но не могут просматривать его исходный код, что значительно усложняет его модернизацию. Разработку этой системы сейчас ведет Mozilla Foundation [5].

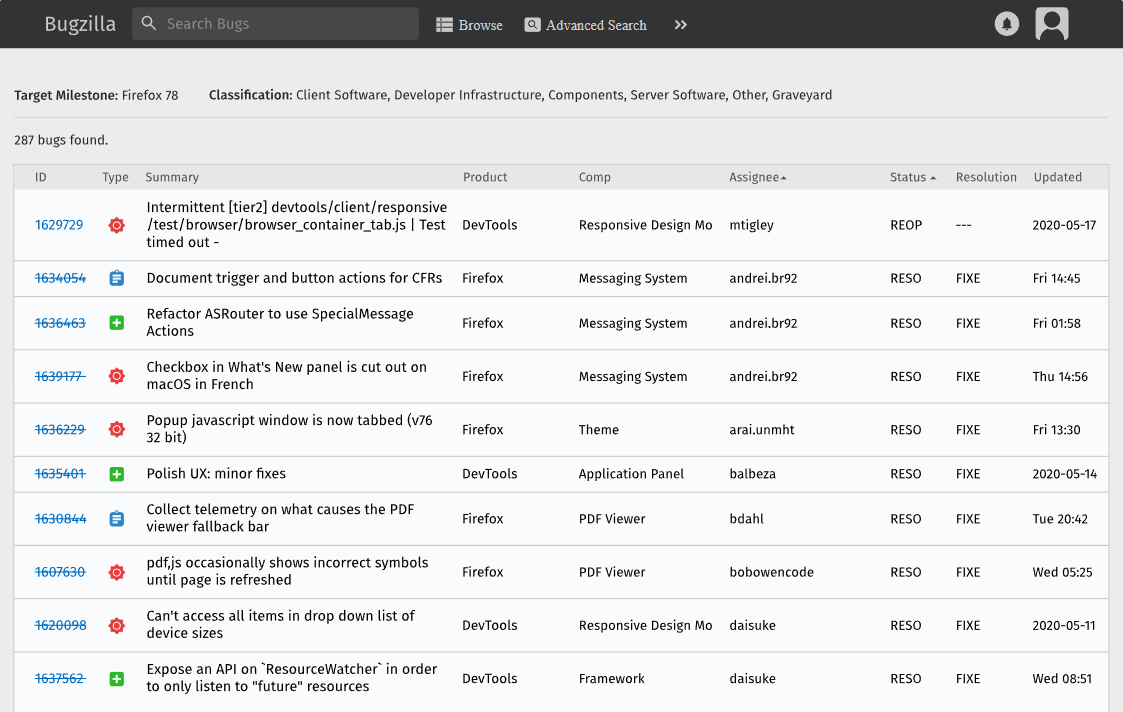


Рисунок 1.2.1 – Внешний вид блока веб-приложения «Bugzilla»

Функционал приложения включает типичный для систем такого рода набор инструментов, необходимых для контроля над ошибками в проектах:

* создание новых дефектов: пользователи могут создавать новые записи о дефектах, указывая приоритет, важность, компонент, версию продукта;
* назначение ответственных: каждому из дефектов может быть присвоен ответственный за его исправление
* отслеживание статуса: каждому из дефектов может содержать информацию об исправлениях и их состоянии;
* создание отчетов: программа может предоставлять отчеты и статистику по дефектам для анализа и принятия управленческих решений;
* поиск и фильтрация: предоставляет возможности поиска и фильтрации дефектов для быстрого доступа к нужной информации;

Однако следует отметить, что установка Bugzilla может оказаться достаточно трудоемкой из-за зависимости от модулей Perl. Также взаимодействие с ПП может быть затруднено из-за сложности администрирования или непонятно интуитивно интерфейса — наиболее четко это прослеживается при пользовании приложением для ОС компьютера (см. рисунки 1.2.1 – 1.2.2) .

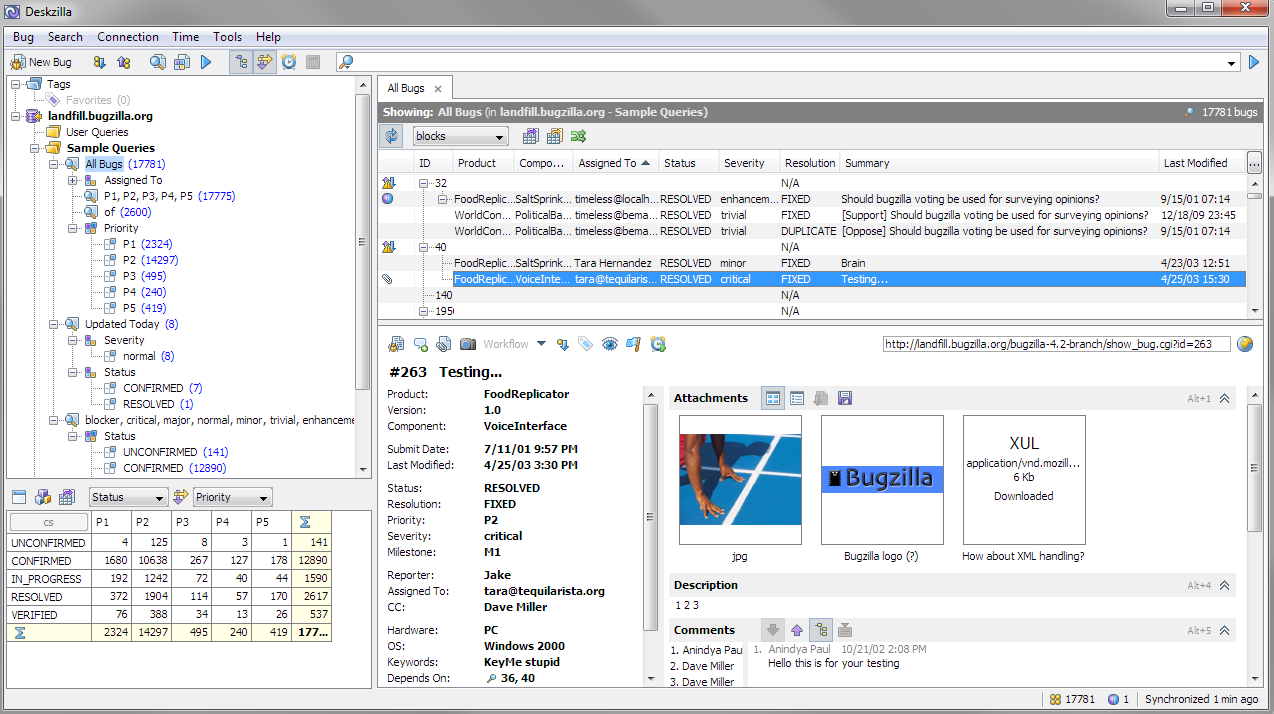


Рисунок 1.2.2 – Внешний вид блока программы «Bugzilla»

Для работы Bugzilla необходимы Apache, Perl и БД MySQL, что требует отдельной настройки и установки и усложняет процесс внедрения самой системы.

2. JIRA – система управления проектами и отслеживания задач, разработанная компанией Atlassian [6]. В настоящее время система отслеживания дефектов JIRA является одной из самых новых и популярных среди представленных. Она обладает наиболее широкой, можно сказать избыточной, функциональностью среди систем отслеживания. В целом JIRA повторяет архитектуру Bugzilla и обладает схожим функционалом. Далее опишем процесс отслеживание дефектов в данной системе:

Создаваясь, отчет о дефекте отправляется ответственному за реализацию данного функционала, создается так называемая «петля разработчик – тестировщик», в которой происходит цикличная переадресация сообщений. Любой из субъектов может перенаправить сообщение по собственному усмотрению.

Каждой существующей проблеме можно поставить приоритет важности, адресовать на себя, добавить комментарий. Это может быть как общий комментарий, так и комментарий направленный одному человеку, это используется, когда ведущий разработчик переадресует сообщение своему коллеге, указывая какую-то техническую подробность, которая нужна только ему. Сообщению можно установить статус «в начале работы», и соответственно указать, когда работы над ним закончены. Система поддерживает возможность создания персонифицированных сообщений. Вся информация может быть предоставлена пользователям в виде временных отрезков, нанесенных на общую временную шкалу (см. рисунок 1.2.3).

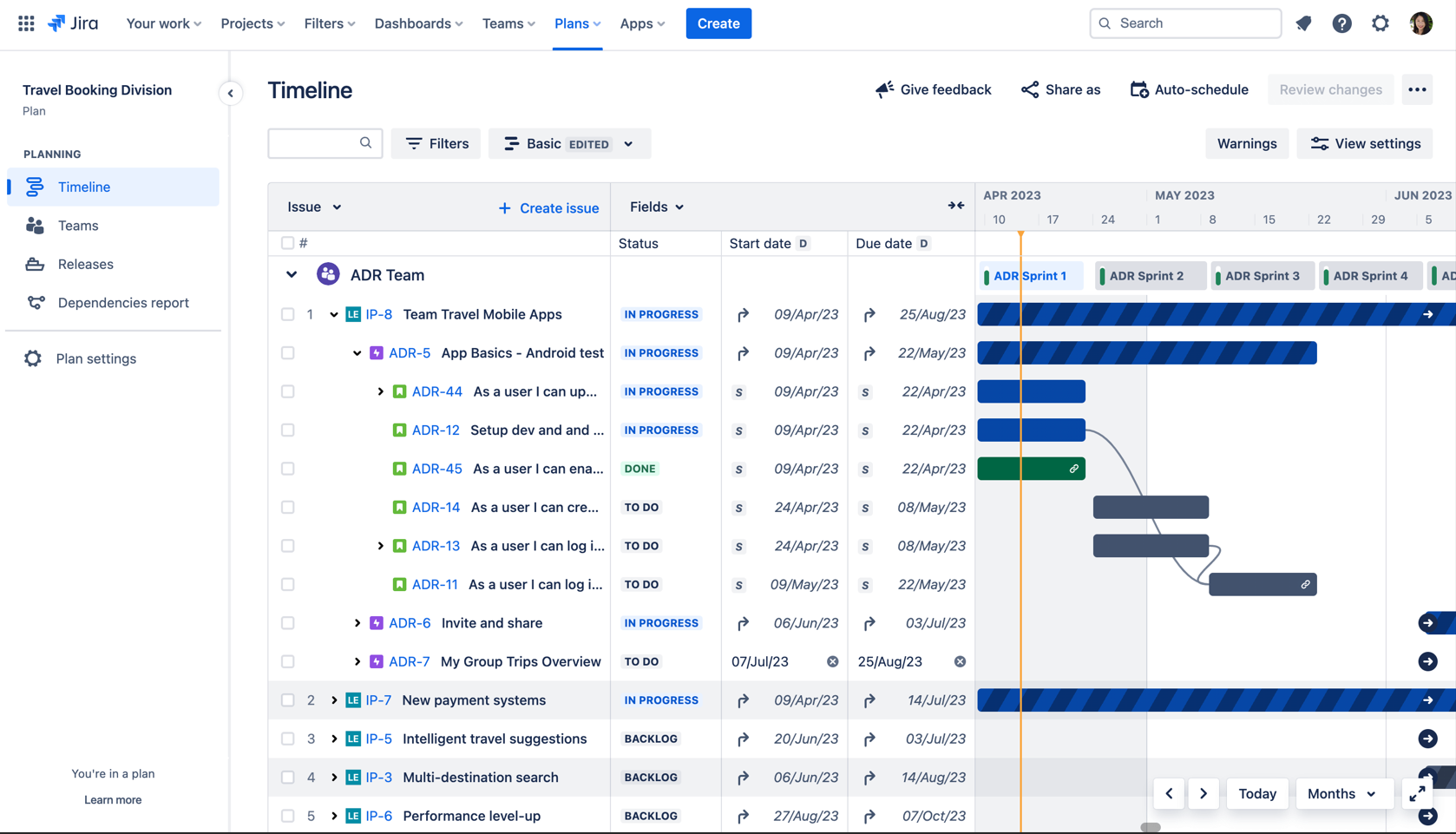


Рисунок 1.2.3 – Внешний вид блока системы «JIRA»

Система JIRA имеет избыток функциональных возможностей, что может быть проблемой для руководителей проектов без достаточного опыта. Вследствие этого сотрудники вынуждены затрачивать значительное количество времени на освоение функционала системы и его разбор.

Система обладает существенным недостатком — она платная. Стоимость установки JIRA на один сервер начинается от 4000 белорусских рублей. Данное обеспечение подходит для крупных компаний с большим штатом тестировщиков и крупным денежным оборотом.

3. Trac – это открытое ПО, являющееся одновременно инструментом для управления проектом и системой отслеживания ошибок. Разработка проекта Trac ведется компанией Edgewall Software и распространяется по лицензии Modified BSD [7], что предоставляет возможность для написания собственных модулей (далее – плагинов) и внедрения их в Trac. Интерфейс Trac представляет собой набор полей и текстовых блоков (см. рисунок 1.2.4), хотя по мнению сообщества он несколько отстает от современных эргономических требований, имеется возможность модернизировать его плагинами из общего доступа или написанными самостоятельно. Система использует в работе SVN репозиторий, так что использовать его имеет смысл только вместе с SVN.

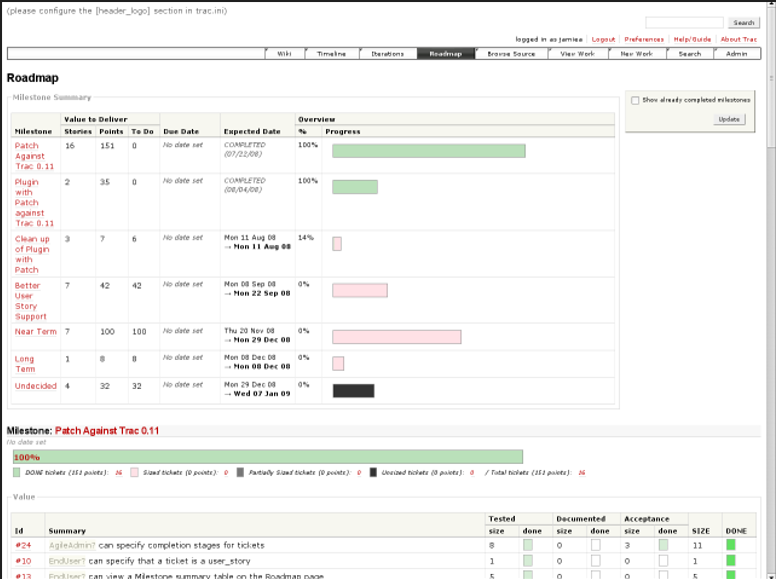


Рисунок 1.2.4 – Внешний вид блока системы «Trac»

Система имеет возможность разделение проекта на этапы контроля и выполнения, все изменения по проекту заносятся на временную шкалу. Trac был написан на Python и является кроссплатформенной системой. Эта система подходит для разработчиков, желающих внедрить комплексное решение для управления проектами и отслеживания ошибок, а так же для тех кто хочет модифицировать систему самостоятельно, дописав к нему дополнительный функционал.

4. The Bug Genie – кроссплатформенная система, написана на PHP. Проект достаточно успешно развивался, последняя версия вышла в марте 2008. The Bug Genie предоставляет базовый набор инструментов для регистрации ошибок, управления приоритетами и формирования задач для разработчиков. Система также предоставляет возможность оповещать всех заинтересованных разработчиков о появлении ошибки.

Система отслеживает ошибки в зависимости от версии и конфигурации ПО. Все ошибки сохраняются в единую БД, представляющую собой базу знаний об ошибках в проекте. Далее по этой БД формируется отчетность (см. рисунок 1.2.5). Однако данное программное решение не позволяет успешно интегрировать себя в окружение, не предусмотренное программистом.

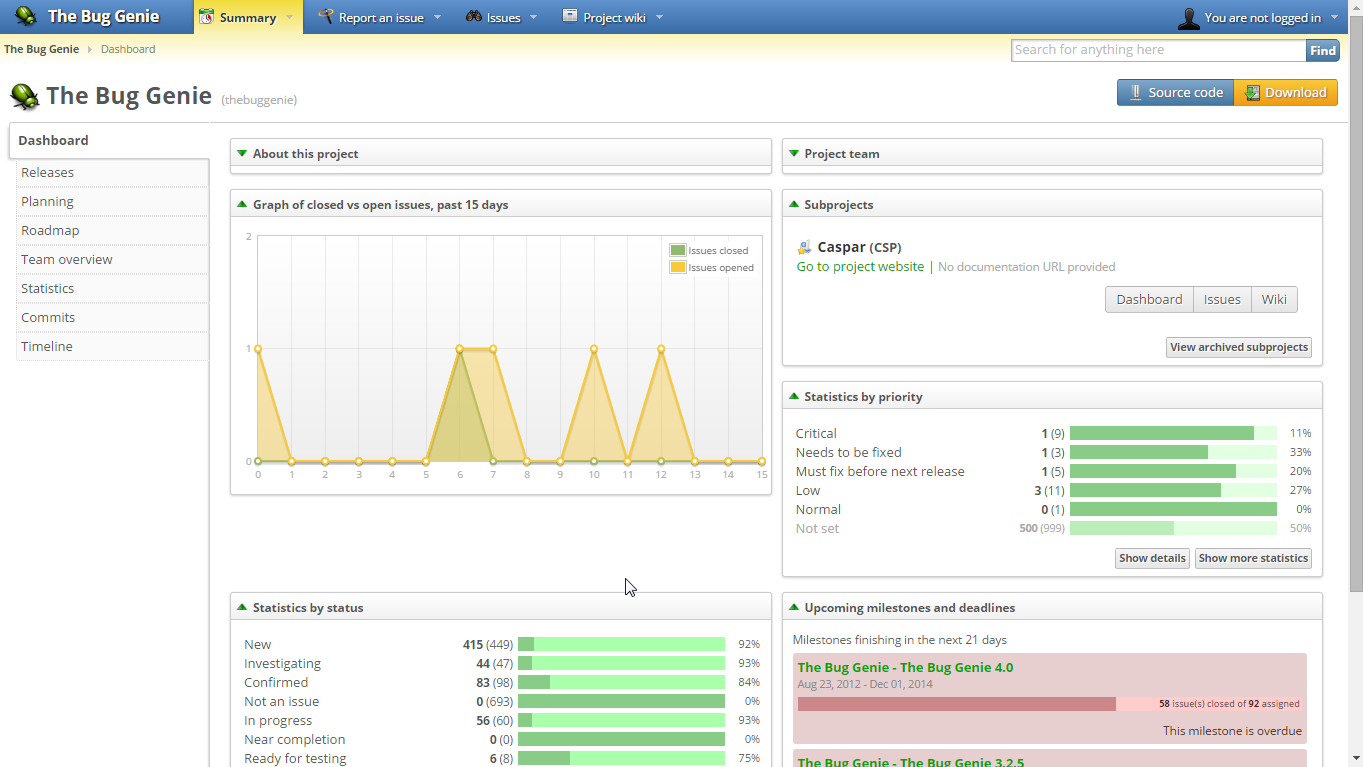
****

Рисунок 1.2.5 – Внешний вид сводки по проекту в «The Bug Genie»

Интеграция с другими инструментами серы разработки является ключевым аспектом при выборе системы отслеживания дефектов. В данном контексте Bugzilla и Trac предлагают относительно ограниченные возможности интеграции по сравнению с JIRA. JIRA, в свою очередь, предлагает широкие возможности интеграции с различными средствами разработки, такими как системы контроля версий (например, Git и менее популярной SVN), инструменты непрерывной интеграции и другими.

Следующим важным критерием является тип лицензии. Bugzilla и Trac являются свободным программным обеспечением, в то время как JIRA является платным. Это может оказать влияние на решение о выборе системы в зависимости от финансовых возможностей компании. Также следует учитывать язык программирования и зависимости каждой системы. Например, Bugzilla требует установки Perl, Apache и MySQL, что может потребовать дополнительных усилий при настройке и поддержке.

Современный эргономичный интерфейс снижает порог вхождения для пользователей и сильно влияет на принятие решения в процессе начального выбора ПП [8], большинство из аналогов СОДПП имеют некоторые затруднения в выдерживании конкуренции при рассмотрении данного аспекта, на что указывает повышенный спрос на платные продукты и доминирующая позиция проекта JIRA.

Интерфейсы систем схожих с разрабатываемой несколько перегружены вследствие специфики предметной области, но уровень перегруженности существенно разнится в зависимости от даты выпуска программы. С течением времени эргономические требования и общие тенденции в дизайне сильно поменялись. Так же от даты выпуска зависит и формат приложения — большинство современного ПО для отслеживания дефектов программных продуктов являются веб-приложениями, или веб-приложениями, которые были интегрированы в ОС компьютера. Такие приложения способны функционировать только при условии стабильного подключения к интернету, так как БД находятся на удаленных серверах.

Освояемость программного продукта тесно связана с его дизайном. Дизайн ПП оказывает прямое влияние на процесс освоения, определяя уровень доступности функционала, ясность интерфейса и интуитивную понятность работы программы. Хорошо спроектированные программные продукты демонстрируют более высокий уровень освояемости, что, в свою очередь, способствует повышению их эффективности использования и удовлетворенности пользователей [9]. Эта характеристика играет ключевую роль в успешной адаптации пользователей к новым программным средствам и определяет скорость достижения оптимальной производительности при их использовании.

Основываясь на приведенных данных, составим таблицу 1.2.1 — таблицу сравнения аналогов СОДПП.

Таблица 1.2.1 – Сравнение аналогов СОДПП

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий | Bugzilla | JIRA | Trac | The Bug Genie |
| Тип лицензии | Свободное ПО | Платное | Открытое ПО | Открытое ПО |
| Язык программирования | Perl | Java | Python | PHP |
| Зависимости | Perl, Apache, MySQL | Java | SVN | PHP |
| Интерфейс | Отстающий | Современный | Отстающий | Устаревший |
| Освояемость | Средняя | Низкая | Низкая | Высокая |
| Сложность установки | Высокая | Средняя | Средняя | Средняя |
| Подходит для маленьких компаний | Нет | Нет | Да | Нет |
| Интеграция с другими инструментами | Возможна | Возможна | Возможна | Ограничена |

(вывод по таблице)

В данном подразделе были приведены лишь некоторые, самые широко известные, ПС по отслеживанию ошибок в программных продуктах. Например, не были затронуты некоторые приложения, которые используются в целях решения задач по отслеживанию дефектов, но не были разработаны для них. Модульные системы, встроенные в какое-либо приложение, целью которого не является отслеживания дефектов (например – GitHub Issues) так же не рассматривались, хотя и пользуются некоторым спросом.

Фактически для отслеживания дефектов может использовать любое существующее средство связи и инструмент, способный содержать в себе несколько вложенных списков — такой функционал может быть реализован даже непрограммно.

Проанализировать эффективность таких видов отслеживания средств невозможно ввиду их специфического назначения и особенностей функционирования. Как уже отмечалось, эти средства могут быть эффективными в определенных сценариях использования, однако их функциональность и возможности могут быть ограничены по сравнению с специализированными инструментами для отслеживания дефектов.

Кроме того, важно учитывать, что эффективность отслеживания дефектов в значительной степени зависит от организации процесса работы команды, внедрения подходящих методологий разработки и использования соответствующих инструментов управления проектом. Даже самая передовая система отслеживания дефектов может оказаться неэффективной, если ее не используют правильно или если в ней отсутствует поддержка и вовлеченность команды.

Таким образом, при выборе средства для отслеживания дефектов необходимо учитывать не только его функциональные возможности, но и контекст использования, особенности проекта и предпочтения команды разработчиков.

****1.3 Выводы и постановка задач на дипломное проектирование****

Рассмотрев существующие аналоги СОДПП и предметную область системы можно обозначить общие пожелания и запросы пользователей к системе, сделать выводы о назначении системы, сформировать ее задачи и функционал.

В частности, из анализа предметной области можно сделать вывод, что СОДПП включает в себя средства налаживания межличностных коммуникаций между тремя основными типами пользователей, разделенных по критерию принадлежности к определенной должности: тестировщик, программист и менеджер проекта. Однако программный продукт в основном используется лишь одним из представленных специалистов — тестировщиком, вследствие чего и будем рассматривать его как рядового пользователя продукта. Существует необходимость выделения другого типа пользователя, способного отслеживать общий прогресс команды разработчиков и вносить правки во все учитываемые ПП — обычно данная возможность необходима менеджеру проекта, но в зависимости от внутреннего распорядка компании может затрагивать и других специалистов. Будем называть такого пользователя «привилегированным», вне зависимости от занимаемой им должности и степени важности для него этой возможности. Итого получаем две роли пользователей в системе:

а) *Рядовой пользователь*: сотрудник отдела тестирования ПО или разработчик, относящийся к реализации части дефектного функционала ПО.

б) *Привилегированный пользователь*: менеджер проекта или человек любой другой должности, ответственный за отчетность о проекте.

Хоть данные роли и имеют очень схожие полномочия в СОДПП, но привилегированный пользователь отличается возможностью наблюдать за деятельностью всех работников, ведущих учет в системе. Разделение на пользователей не является обязательным на этапе предоставления базового программного решения, а в основном необходимо для демонстрации возможности дальнейшего разделения функционала и примера внедрения такового. Базовое решение обычно ограничивается минимальным необходимым функционалом. Выделим его в рамках разрабатываемой системы и коротко опишем каждую из функций системы:

1. Регистрация аккаунта — выполнение процедуры, при которой пользователь заполняет электронную форму, содержащую поля "Имя пользователя" и "Пароль";
2. Авторизация — идентификация и аутентификация существующего пользователя;
3. Добавление ПП — добавление ПП в личный список для дальнейшего учета. Присваивание ПП названия и существующих дефектов данного продукта;
4. Архивирование ПП — сокрытие ПП из общего списка отслеживаемых продуктов и помещение их в отдельный список с разрешения привилегированного пользователя.
5. Редактирование ПП — редактирование названия ПП, текстового описания, список работников, ответственных за исправление дефекта;
6. Добавление дефектов — установка названия дефекта, его важности, приоритета, результата, ожидаемого результата, описания найденного дефекта, комментария составителя и статуса;
7. Изменения состояния существующего дефекта — пометка дефекта как исправленного или добавление его в архив;

Следует отметить, что у пользователей СОДПП могут быть несколько завышенные требования к системе, ввиду своей сильной включенности в сферу программного обеспечения и углубленного понимания общих практик (стандартов) в индустрии его разработки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Андрианов, М. С. Невербальная коммуникация / Андрианов М.С. — Москва: Институт Общегуманитарных Исследований, 2007 — С. 91 – 92

2 Когаловский, М. Р. Энциклопедия технологий баз данных / Когаловский М. Р. — Москва: Финансы и статистика, 2002 — 800 с.

3 Состояние открытого исходного кода. Статистика [Электронный ресурс] — Режим доступа: https://github.blog/2023-11-08-the-state-of-open-source-and-ai

4 Фонд Свободного ПО (FSF) [Электронный ресурс] — Режим доступа: https://www.gnu.org/philosophy/free-sw.html

5 Bugzilla: документация проекта [Электронный ресурс] — Режим доступа: https://www.bugzilla.org/docs/4.2/en/html

6 Atlassian: документация проекта [Электронный ресурс] — Режим доступа: https://www.atlassian.com/legal/software-license-agreement

7 Trac: документация проекта [Электронный ресурс] — Режим доступа: https://trac.edgewall.org/wiki/TracGuide

8 Котлер, Ф. Маркетинг, менеджмент: анализ, планирование, внедрение, контроль / Ф. Котлер; пер. с англ. — СПб.: Питер, 1999 — 896 с.

9 ???? ОСВОЯЕМОСТЬ ЭТО ВАЖНО ????