УДК 004.054

АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ КЛИЕНТСКОЙ КОНФИГУРАЦИИ ПРОДУКТОВ КОМПАНИИ ONEVIZION

Токмагашева Ю. В., магистрант гр.ПИм-171, II курс
Пилецкая А. Б., магистрант гр.ПИм-171, II курс
Емельянов И.Д., магистрант гр.ПИм-171, I курс
Научный руководитель: Пимонов А. Г., д. т. н., профессор
Кузбасский государственный технический университет имени Т.Ф. Горбачева
г. Кемерово

Управление сложными и географически рассредоточенными программами может представлять основные операционные проблемы для компании. Слишком часто информация, необходимая для принятия обоснованных и своевременных решений, разбросана по нескольким корпоративным системам, содержится в электронных письмах или похоронена в специальных электронных таблицах [5].

OneVizion – это высокопроизводительная прикладная платформа как услуга (high productivity application Platform as a Service – hpaPaaS), которая решает сложную проблему централизации наборов многоуровневой информации на предприятиях. OneVizion позволяет уверенно принимать решения путем плавной интеграции активов, документов и данных с финансовыми, плановыми и операционными системами с помощью гибких и масштабируемых приложений, которые отвечают постоянно меняющимся потребностям бизнеса [1].

Onevizion Trackor – гибкий, масштабируемый инструмент управления информацией. Такие характеристики достигаются за счет того, что для каждого клиента конфигурация системы настраивается индивидуально и полностью адаптируется к его уникальным бизнес-потребностям, рабочим процессам и наборам данных. Кроме того, пользователям доступны такие функции, как вывод данных в графическом виде – графики, карты, последовательность шагов для автоматизации бизнес-процессов, также доступны различные отчеты, импорт/экспорт данных и другие полезные функции, которые настраиваются для клиентов индивидуально [1].

Иерархическая структура данных платформы – Trackor Tree – фиксирует любые исключения и соответствует уникальной организации и требованиям клиента. Данные, конвертируемые в стандартизированные объекты – Trackors – предоставляют руководящим группам беспрецедентную видимость из единого источника в режиме реального времени [1].

Динамический API OneVizion обеспечивает плавную интеграцию с важными устаревшими инструментами, используемыми клиентом ранее, и другими приложениями уровня предприятия, такими как финансовые или бухгалтерские системы [1].

Обеспечить защиту данных в системе и проконтролировать санкционированность действий пользователей позволяет жестко организованное разграничение прав доступа. Механизмы управления доступом субъектов к объектам доступа строятся на концепции единого диспетчера доступа. Сущность этой концепции состоит в том, что диспетчер доступа (монитор ссылок) – выступает посредником-контролером при всех обращениях субъектов к объектам.

Для назначения прав доступа пользователей, или, другими словами, привилегий пользователей, используются Security Roles. Различные Security Roles позволяют администратору определять доступ пользователя для просмотра и изменения данных в системе OneVizion.

В OneVizion используется двухнедельный цикл выпуска новых версий. Для обеспечения бесперебойной работы промышленной версии конфигурации клиента сначала идет обновление так называемой UAT-версии (версии конфигурации для проведения приемочного пользовательского тестирования). Далее ведется активное тестирование данной версии, выявление и исправление возникших в ходе тестирования ошибок, после чего выполняется обновление промышленной версии – PROD-версии.

Каждую ночь запускаются автоматизированные тесты, которые проверяют корректность работы критичных, по мнению разработчиков, функций. Данная возможность реализована при помощи Jenkins – инструмента для непрерывной интеграции программного обеспечения, который имеет открытый исходный код и предоставляет большое количество плагинов для поддержки создания, развертывания и автоматизации проектов [4].

Запускаемые тесты представляют собой имитацию действий пользователя над графическим интерфейсом платформы в веб-браузере. Данные тесты написаны на объектно-ориентированном языке программирования – Java, имитируемая тестами работа пользователя реализуется с использованием инструмента для автоматизации действий веб-браузера – Selenium [2].

Значимость вариантов использования системы для пользователей разная, и то, что для одних может стать незначительным, для других окажется критичным. Как правило, клиенты уделяют недостаточно внимания тестированию UAT-версии, не желая тратить или не имея ресурсов на это, в результате чего некоторые ошибки обнаруживаются только в результате опытной эксплуатации системы, замедляя работу пользователей или делая ее вовсе невозможной. После возникновения таких ошибок клиенты не соглашаются обновляться на новые версии, это приводит к тому, что разработчикам приходится поддерживать одновременно несколько версий продукта, в следствии чего количество работы у них увеличивается (исправление ошибок ведется не только в текущей версии, но и вносится на другие поддерживаемые версии, а, значит, меньше времени остается на разработку и доработку и т. д.).

В связи с этим, возникает необходимость доработать процесс тестирования – сделать так, чтобы воспроизводились критичные для конкретного клиента Use-Cases (варианты использования системы) еще до выхода версии в PROD и была возможность исправить возникающие при воспроизведении

Use-Cases ошибки, тем самым обеспечивая наиболее безболезненный переход на новую версию.

Первоочередной задачей при решении данной проблемы является организация централизованного сбора и хранения критичных для конкретного клиента Use-Cases, для реализации которой был разработан модуль конфигурации, отвечающий за это и называемый Test-Cases (рис. 1).

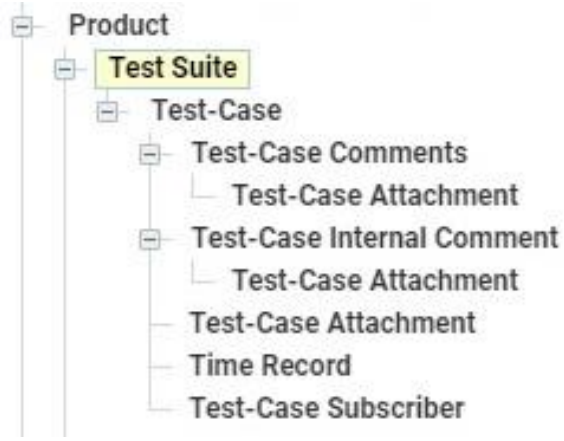

Рисунок 1 – Структура модуля Test-Cases

В данном модуле доступны такие функции, как создание и редактирование Use-Case (рис. 2), описание ожидаемого после его выполнения результата, прикрепление нужных файлов, добавление и просмотр комментариев для обсуждения данного Use-Case, создание списка оповещаемых об изменении use-case, добавлении новых комментариев.

Далее необходимо разработать автоматизированные тесты для выполнения описанных в Use-Cases действий и оповестить о результатах тестирования и об ошибках, которые возникают в ходе выполнения этих Use-Cases автоматизированными тестами.

Так как имеющиеся автоматизированные тесты написаны на объектно-ориентированном языке программирования Java и используется инструмент Selenium, а также имеется отлаженный механизм запуска тестов и оповещения о результатах их выполнения, было решено продолжить использование данных инструментов и средств. Кроме того, уже имеется написанный API, предоставляющий возможность использования готовых методов, реализующих имитацию однотипных действий пользователя, таких как вход в систему, открытие формы, переход в определенный пункт меню системы и т. д., что значительно упрощает процесс написания автоматизированных тестов по критичным для клиента Use-Cases.

Учитывая то, что для каждого клиента критичные Use-Cases разные, соответственно, написанные по этим Use-Cases автоматизированные тесты тоже имеют свои нюансы, хранение автоматизированных тестов для каждого клиента организовано в отдельном репозитории GitHub – веб-сервиса, основанного на контроле версий [3]. Также для каждого клиента в Jenkins для запуска этих тестов создан отдельный Job – регулярно выполняемая задача.

Процесс запуска и выполнения автоматизированных тестов происходит следующим образом (рис. 3) – клонируется репозиторий из GitHub, в котором хранится проект с автоматизированными тестами, происходит компиляция этого проекта, затем запуск тестов, после чего результаты выполнения тестов отправляются на электронную почту.

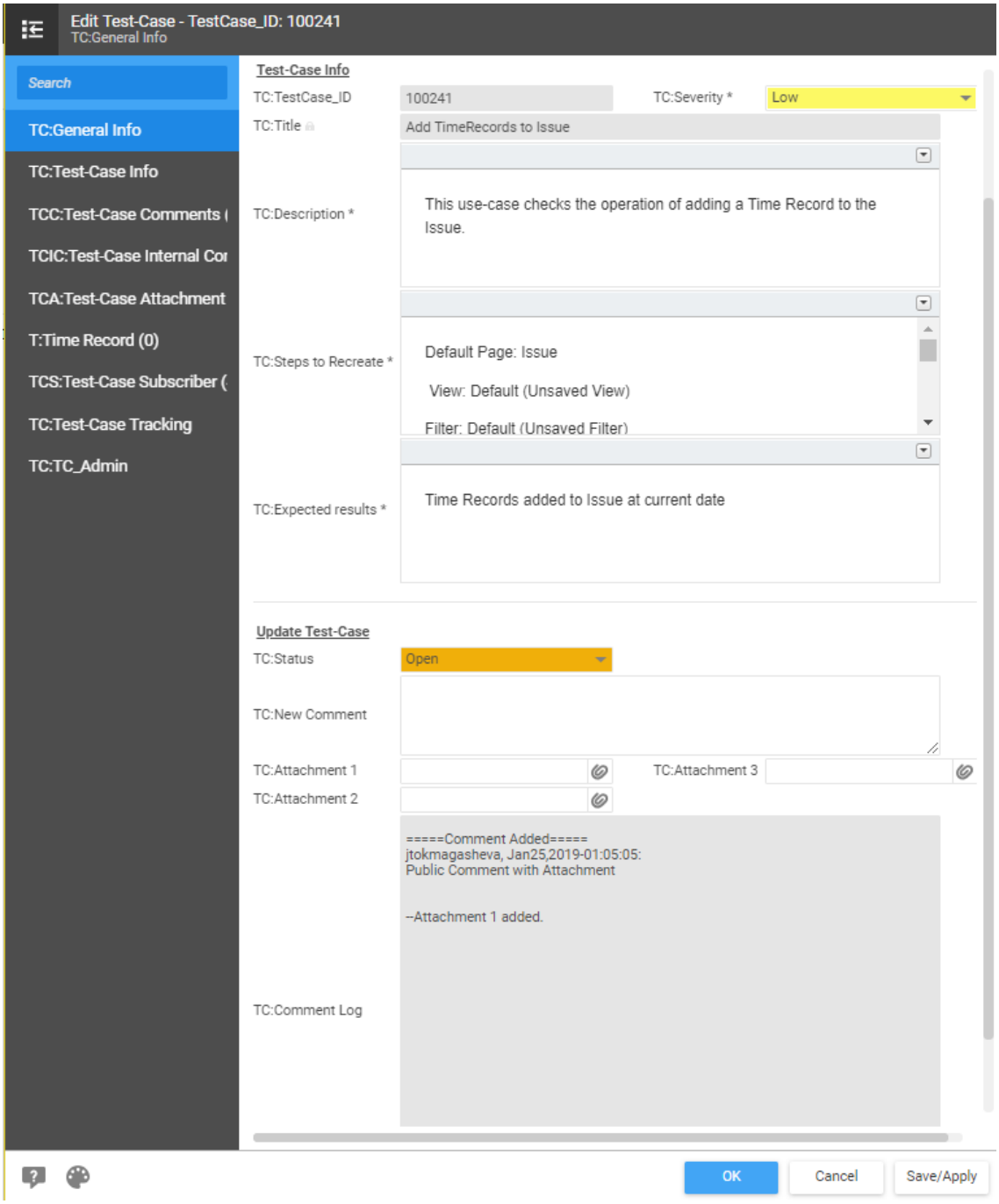

Рисунок 2 – Форма редактирования Test-Case

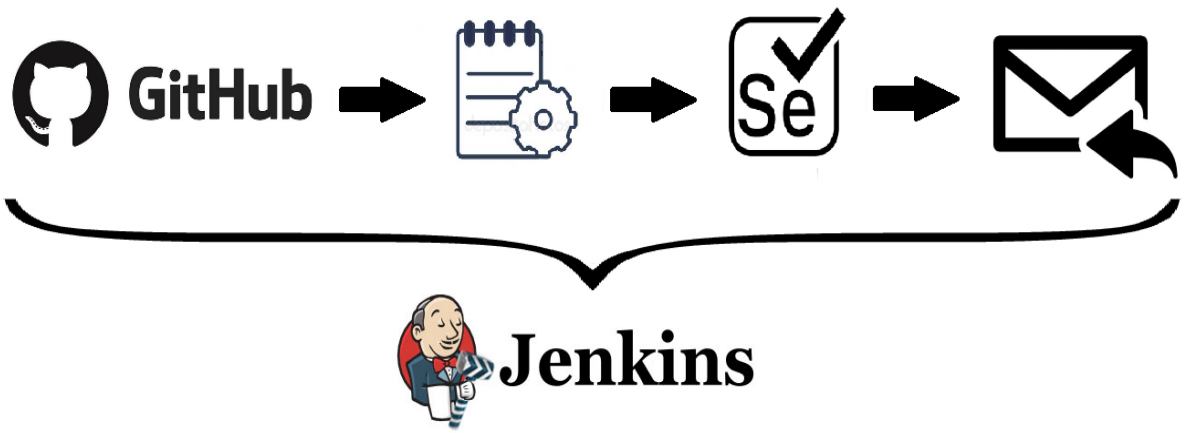

Рисунок 3 – Схема процесса запуска и выполнения автоматизированных тестов, написанных по критичным для клиентов Use-Cases

В результате проделанной работы были решены следующие задачи:

1. организован централизованный сбор и хранение критичных для конкретного клиента Use-Cases;
2. разработаны автоматизированные тесты по предоставленным клиентами критичными для них Use-Cases;
3. настроены регулярный запуск разработанных для клиентов автоматизированных тестов и оповещение о результатах их выполнения.

Выполнение вышеперечисленных задач дало возможность клиентам проводить более тщательное тестирование критичных для них моментов работы платформы до того, как обновится их промышленная версия, тем самым позволяя поддерживать стабильную бесперебойную работу клиентской конфигурации платформы.

Список литературы:

1. OneVizion Product Documentation // OneVizion Simply Smarter Information Management [Электронный ресурс]. – Режим доступа: https://onevizion.atlassian.net/wiki/spaces/WIKI/overview? mode=global, свободный (дата обращения: 20.03.2019).
2. Что такое Selenium? [Электронный ресурс]. – Режим доступа: https://selenium2.ru/, свободный (дата обращения: 18.03.2019).
3. Git и GitHub: что это такое и в чем разница // Tproger [Электронный ресурс]. – Режим доступа: https://tproger.ru/translations/difference-between-git-and-github/, свободный (дата обращения: 17.03.2019).
4. Главная страница сайта //Jenkins [Электронный ресурс]. – Режим доступа: https://jenkins.io/, свободный (дата обращения: 20.03.2019).
5. Степанюк, А. В. Исследование методов тестирования, проектирование и разработка автоматизированных тестов для графического интерфейса с помощью инструмента Selenium: диссертация магистра прикладной информатики, Кузбасский государственный технический университет имени Т. Ф. Горбачева, Кемерово, 2018.