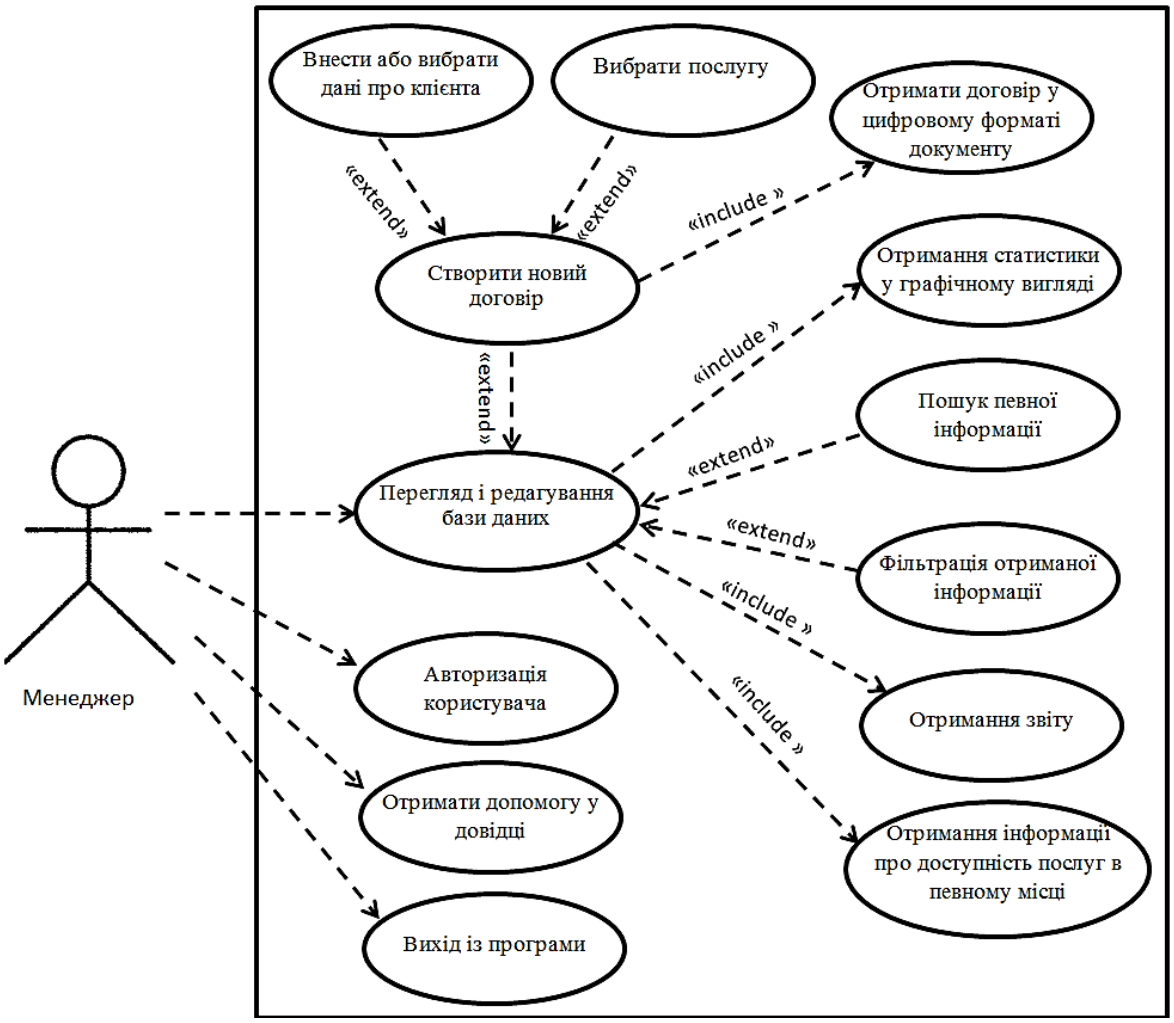

Рисунок 1 – Діаграма прецедентів «АРМ менеджеру інтернет-провайдера»

СПИСОК ЛІТЕРАТУРИ

1. Мишенин А. И. Автоматизированные рабочие места управленческого аппарата / А. И. Мишенин. – 1999.

2. Мескон М. Автоматизированные рабочие место : пер. с англ. / М. Мескон, М. Альберт, Ф. Хедоури ; общ. ред. Л. И. Евенко. – М. : Дело, 1994.

СОВРЕМЕННЫЕ СИСТЕМЫ ДЛЯ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ WEB-СЕРВИСОВ И ПРИКЛАДНЫХ ПРИЛОЖЕНИЙ

Кононенко А. А., Абдулов А. Р.

ДГМА, г. Краматорск

Самым первым шагом успешного теста той или иной задачи является правильная постановка даты релиза current-версии. Необходимо тщательно продумать объемы тестирования и всевозможные проблемные моменты, которые смогут остановить работу QA (от англ. quality assurance –

обеспечение качества). Так, например, случайно разработчики зацепили одну из механик, хотя совсем ее не трогали. Нужно оценить количество сделан-ных задач, т. е. тех, которые программисты и дизайнеры успели отдать тестировщикам на проверку, а также просмотреть сложность и объемы невыполненных задач. Также необходимо понять, сколько новых механик появится в текущей версии. Это самое важное, поскольку в них чаще всего скрывается множество багов, для воспроизведения которых понадобится очень много времени.

Тестирование необходимо для того, чтобы гарантировать устойчивость программного продукта в любых обстоятельствах. Само понятие означает соответствие имеющегося программного продукта ожидаемому результату. Только этот этап делится на четыре основных «титана» успешной проверки. Первый шаг, который был описан выше, – планирование. После этого создаются тест-кейсы и тест-план (тест-дизайн), для выявления наиболее опасных мест программы, которые приводят к тем или иным багам (тест-кейсы должны сортироваться по важности, начиная от blocker-ошибок, и заканчивая trivial-дефектами). Следующим шагом является само тестирование. Тестироваться будут функциональные, нефункциональные и связанные с обновлениями критерии.

При функциональном тестировании будет проверяться взаимодействие с пользователем (реагирование персонажа на указания игрока), а также конфиденциальность (сокрытие личной информации от мошенников) и способность взаимодействия с компонентами и системами (кросс-браузерность, кроссплатформенность и разрешениями экрана). Предполагается написание тест-кейсов.

Нефункциональное тестирование:

- Максимальное количество пользователей, использующих игру одновременно.
- Скорость выполнения операций.
- Изменение производительности при увеличении одновременного использования игры несколькими пользователями.
- Работа с большими объемами операций.
- Скорость восстановления приложения после интенсивных нагрузок.
- Изменение работоспособности приложения при многочасовой работе.
- Удобство установки.
- Проверка корректности работы после установки.
- Легкость при использовании приложения.
- Хороший эмоциональный посыл.
- Скорость восстановления приложения после сбоя.
- Работа приложения на платформах.

Тестирование обновлений предполагает корректность работы после обновлений. После всего этого переходим к этапу анализа полученных результатов.

Общая работоспособность часто уходит на задний план, поскольку все внимание уделяется вновь появившимся задачам (особенно, если их много). К сожалению, такой подход приводит к проскальзыванию багов на сервера пользователей, приводящим к failure. Но, как говорится, «Хорошего тестировщика баги любят». Опытные QA Engineers обладают намеченным глазом на места скрытых багов.

Для оценки общей работоспособности игры, необходимо:

- Проверка установки (если это необходимо) или же запуск (в браузерных играх).
- Проверка корректной работы и отображения в различных браузерах (для онлайн игр).
- Проверка адекватной работы на платформах, кроссплатформенность.
- Способность быстрого восстановления после сбоев.
- Способность стабильно работать при долгом использовании игры.

После этого тестируется геймплей. На примере игры Farland тестируем:

- Монетизация.
- Создание персонажа.
- Передвижение персонажа по игровому полю.
- Очистка местности от камней и растительности.
- Сбор материалов.
- Выполнение квестов.
- Строительство зданий и вспомогательных объектов.
- Осваивание новых территорий.
- Посещение земель друзей.

Приоритеты в тестировании расставляются для того, чтобы можно было составить план работы. Те элементы, которые требуют наискорейшего исправления ошибок, обладают самым высоким приоритетом. Чем ниже приоритет, тем позже будет тестироваться объект.

Приоритеты расставляются в зависимости от тяжести ошибок и их способности оттолкнуть пользователя.

Так, первыми проверяются ошибки, которые могут привести к поломке игры, а именно к нарушению ее функциональности (например, отсутствие кроссбраузерности и кроссплатформенности, способность к беспроблемному восстановлению и устойчивости к сбоям). Это может быть проверка сборки и установки.

Далее решаются проблемы с ошибками в бизнес-логике, а также такими, которые временно могут привести приложение в нерабочее состояние. Тестирование работы в больших нагрузках сети (одновременная работа большого количества пользователей, многочасовая работа приложения, значительное увеличение данных в базе приложения) и при внезапном отключении электричества.

После этого проверяются части, в которых бизнес-логика написана корректно и не вызывает проблем, но интерфейс приложения оставляет желать лучшего. Это может быть тестирование функций взаимодействия с пользователем (в том числе удобство работы с игрой) и отклика на его действия, а также взаимодействие нескольких компонентов самой игры друг с другом.

По окончании работы с этими ошибками следует приступить к работе над теми ошибками, которые не мешают работе проекта и малозаметны для пользователей. Это больше «косметический» этап, чтобы сделать продукт идеальным.

Часто бывает, что приходится уменьшить затраты времени на проверку. Например, несвоевременное возвращение запланированных тасок в QA, или же совмещение двух версий (что чаще всего исходит из первого примера). Для этого существует такое понятие, как тест-дизайн.

Тест-дизайн – это один из этапов тестирования, который представляет собой написание тест-кейсов. Они позволяют протестировать весь функционал в режиме пользовательской игры. Это весьма кропотливый труд, но он позволяет найти проколы игры и исправить их.

- Проверяется наличие багов при граничных значениях (например, размещение жилища, цепляя уже существующий объект на игровом поле)
- Минимизация количества тестов путем совмещения похожих тест-кейсов (например, невозможность постройки жилища и посадки растений на крыше другого дома).
- Минимизация количества тестов путем объединения тест-кейсов с одинаковым откликом игры (например, при размещении жилища или растений на другом объекте или в заблокированной местности, приложение выдаст ошибку).
- Пред усмотрение действий игрока, а также возможных ошибок.
- Проверка переходов в различные состояния приложения, что покажет возможные ошибки (например, переход к другому игроку во время строительства жилища или уборки деревьев).

ПРАКТИЧНІ РЕКОМЕНДАЦІЇ ЩОДО РОЗРОБКИ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ ГРИ «МОРСЬКИЙ БІЙ»

Копецький Я. Р., Гетьман І. А.
ДДМА, м. Краматорськ

«Морський бій» – це гра для двох людей, в якій гравці по черзі називають координати на невідомій їм карті суперника. Якщо у суперника в тому місці є корабель(координати зайняті), то корабель або його частина «тоне», а той, хто влучив здобуває право ходу. Мета гравця – першим потопити всі кораблі супротивника. Гру вперше випустила у вигляді настільної гри компанія Milton Bradley Company 1931 року [1].