МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное

Учреждение высшего образования

**«Гжельский государственный университет»** (ГГУ)

Колледж ГГУ

Специальность 09.02.07 Информационные системы и программирования

**Реферат**

**по дисциплине «МДК 01.01 Технология разработки программного обеспечения »**

**на тему « Разработка тестового сценария»**

ВЫПОЛНИЛА:

Студентка группы ИСП-О-17

Янгабозова Анастасия

ПРОВЕРИЛА:

Прокуронова А.Ю.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

п. Электроизолятор

2019 г.

Для разработки тестовых сценариев и выполнения тестов используются системы управления тестированием, существенно повышающие производительность тест-дизайнеров и тестировщиков, а также обеспечивающие видимость уровня качества приложений среди всех участников проекта.

Тестовые сценарии неразрывно связаны с требованиями, изменения в которых должны своевременно отражаться в тестовой документации, что позволяет сделать система управления жизненным циклом разработки приложений, при помощи механизма трассировок.

При выполнении теста тестировщик отмечает результат прохождения одного шага или всего тестового сценария, прикрепляет обнаруженные ошибки и другую вспомогательную информацию: скриншоты, дампы, логи и т.п.

**Тестовые сценарии удобно объединять в тест-планыпо назначению:**

· тестирование релиза, то есть очередной версии продукта;

· тестирование развертывания;

· тестирование удобства использования;

· конфигурационное тестирование;

· тестирование безопасности и т.п.

Зачастую ручное тестирование превращается в рутину и занимает значительное время, что отрицательно сказывается на скорости выпуска релизов.

**Автоматизация тестирования позволяет:**

· высвободить ресурсы для проведения более сложных видов тестирования;

· снизить количество дефектов, доходящих до стадии контроля качества;

· ускорить выпуск релизов.

Сведение результатов автоматических и ручных тестов в системе управления качеством, позволяет всем участникам проекта видеть уровень качества очередного релиза, контролировать его изменение и опираться на эту информацию при планировании своей работы.

**Перед получением результата программа проходит несколько уровней:**

· Тестирование компонентов — тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция. Часто тестирование компонентов осуществляется разработчиками программного обеспечения.

· Интеграционное тестирование — тестируются интерфейсы между компонентами, подсистемами или системами. При наличии резерва времени на данной стадии тестирование ведётся итерационно, с постепенным подключением последующих подсистем.

· Системное тестирование — тестируется интегрированная система на её соответствие требованиям.

· Альфа-тестирование — имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком. Чаще всего альфа-тестирование проводится на ранней стадии разработки продукта, но в некоторых случаях может применяться для законченного продукта в качестве внутреннего приёмочного тестирования. Иногда альфа-тестирование выполняется под отладчиком или с использованием окружения, которое помогает быстро выявлять найденные ошибки. Обнаруженные ошибки могут быть переданы тестировщикам для дополнительного исследования в окружении, подобном тому, в котором будет использоваться программа.

· Бета-тестирование — в некоторых случаях выполняется распространение предварительной версии (в случае проприетарного программного обеспечения иногда с ограничениями по функциональности или времени работы) для некоторой большей группы лиц с тем, чтобы убедиться, что продукт содержит достаточно мало ошибок. Иногда бета-тестирование выполняется для того, чтобы получить обратную связь о продукте от его будущих пользователей.

Часто для свободного и открытого программного обеспечения стадия альфа-тестирования характеризует функциональное наполнение кода, а бета-тестирования — стадию исправления ошибок. При этом как правило на каждом этапе разработки промежуточные результаты работы доступны конечным пользователям.

**Пример:**  
Перем КонтекстЯдра;

Перем Ожидаем;

Процедура Инициализация(КонтекстЯдраПараметр) Экспорт

КонтекстЯдра = КонтекстЯдраПараметр;

Ожидаем = КонтекстЯдра.Плагин("УтвержденияBDD");

КонецПроцедуры

Процедура ЗаполнитьНаборТестов(НаборТестов) Экспорт

НаборТестов.НачатьГруппу("Сценарные тесты с вызовом деструктора группы", Истина);

НаборТестов.Добавить("КонструкторСценария", "Начало сценария");

НаборТестов.Добавить("ТестДолжен\_ПроверитьЖурналВыполнения", "Проверяю журнал выполнения шагов");

НаборТестов.ДобавитьДеструктор("ДеструкторСценария", "Обязательное завершение сценария");

КонецПроцедуры

// { Сценарный тест

Процедура КонструкторСценария() Экспорт

ЖурналВыполнения = "КонструкторСценария;";

КонтекстЯдра.СохранитьКонтекст(ЖурналВыполнения);

КонецПроцедуры

Процедура ТестДолжен\_ПроверитьЖурналВыполнения() Экспорт

ЖурналВыполнения = КонтекстЯдра.ПолучитьКонтекст();

Ожидаем.Что(ЖурналВыполнения, "ТестДолжен\_ПроверитьЖурналВыполнения").Равно("КонструкторСценария;");

ЖурналВыполнения = ЖурналВыполнения + "ПроверитьЖурналВыполнения;";

КонтекстЯдра.СохранитьКонтекст(ЖурналВыполнения);

КонецПроцедуры

Процедура ДеструкторСценария() Экспорт

ЖурналВыполнения = КонтекстЯдра.ПолучитьКонтекст();

Ожидаем.Что(ЖурналВыполнения, "ДеструкторСценария").Равно("КонструкторСценария;ПроверитьЖурналВыполнения;");

ЖурналВыполнения = ЖурналВыполнения + "ДеструкторСценария;";

КонтекстЯдра.СохранитьКонтекст(ЖурналВыполнения);

КонецПроцедуры

// }