НАУЧНЫЙ ЖУРНАЛ
НАУКА И МИРОВОЗЗРЕНИЕ

СРАВНИТЕЛЬНОЕ ИССЛЕДОВАНИЕ МЕТОДОВ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Аррыкова Гульджемал Керимназаровна

Старший преподаватель Международного университета нефти и газа имени Ягшыгелди Какаева, г. Ашхабад Туркменистан

Гелдиев Сердар Отузбаевич

Преподаватель Международного университета нефти и газа имени Ягшыгелди Какаева, г. Ашхабад Туркменистан

Ходжамырадова Гулсенем Реджепмырадовна

Студент Международного университета нефти и газа имени Ягшыгелди Какаева, г. Ашхабад Туркменистан

Ходжаева Айгозел Магсадовна

Студент Международного университета нефти и газа имени Ягшыгелди Какаева, г. Ашхабад Туркменистан

Аннотация.
Тестирование программного обеспечения предполагает процесс, используемый для измерения качества разработанного компьютерного программного обеспечения. В нем представлены все ошибки, просчеты и огрехи в разработанном программном обеспечении. В этой статье описываются и сравниваются три наиболее распространенных и часто используемых метода тестирования программного обеспечения для обнаружения ошибок: тестирование белого ящика, тестирование черного ящика и тестирование серого ящика.
Ключевые слова: черный ящик, демонстрация, обнаружение, серый ящик, тестирование программного обеспечения, белый ящик

ВВЕДЕНИЕ

Тестирование программного обеспечения является важным видом деятельности в жизненном цикле разработки программного обеспечения. Тестирование программного обеспечения - это процесс оценки функциональности и корректности программы посредством выполнения или анализа. Тестирование программного обеспечения является важным средством оценки программного обеспечения для определения его качества. Поскольку тестирование обычно отнимает 40 ~ 50% усилий по разработке и требует больше усилий для систем, требующих более высокого уровня надежности, оно является важной частью разработки программного обеспечения. С развитием языков четвертого поколения (4GL), которые ускоряют процесс внедрения, доля времени, отводимого на тестирование, увеличилась.

Тестирование программного обеспечения - это не “серебряная пуля”, которая может гарантировать производство высококачественных программных систем.

В то время как “правильное” доказательство корректности демонстрирует, что программная система (которая точно соответствует ее спецификации) всегда будет работать заданным образом, тестирование программного обеспечения, которое не является полностью исчерпывающим, может только предполагать наличие недостатков и не может доказать их отсутствие. Более того, невозможно полностью протестировать приложение, потому что область входных данных программы слишком велика, существует слишком много возможных путей ввода и проблемы с дизайном и спецификациями трудно поддаются тестированию. Первый и второй пункты представляют очевидные сложности, а последний пункт подчеркивает трудность определения того, являются ли спецификация решения проблемы и дизайн его реализации также правильными.

По мере увеличения объема технического обслуживания и модернизации существующих систем также потребуется значительное количество тестов для проверки систем после внесения изменений. Несмотря на достижения в области формальных методов и техник верификации, система все еще нуждается в тестировании перед ее использованием. Тестирование остается действительно эффективным средством обеспечения качества программной системы нетривиальной сложности, а также одной из самых запутанных и наименее понятных областей разработки программного обеспечения. Тестирование, важная область исследований в области компьютерных наук, вероятно, станет еще более важной в будущем.

ЦЕЛЬ ТЕСТИРОВАНИЯ

Цель тестирования - найти проблемы и устранить их для улучшения качества (рис.1). Тестирование программного обеспечения обычно составляет 40% бюджета на разработку программного обеспечения.

Существует четыре основные цели тестирования программного обеспечения:

1. Демонстрация: он демонстрирует функции в особых условиях и показывает, что продукты готовы к интеграции или использованию.
2. Обнаружение: он обнаруживает дефекты, ошибки и недоработки. Он определяет возможности и ограничения системы, качество компонентов, рабочие продукты и саму систему.
3. Предотвращение: он предоставляет информацию для предотвращения или уменьшения количества ошибок, уточняет системные спецификации и производительность. Определите способы избежать риска и проблем в будущем.
4. Повышение качества: Проводя эффективное тестирование, мы можем свести к минимуму ошибки и, следовательно, улучшить качество программного обеспечения.

Рисунок 1. Поток тестовой информации

РАЗЛИЧНЫЕ МЕТОДЫ ТЕСТИРОВАНИЯ

А. Тестирование черного ящика

Тестирование черного ящика основано на спецификациях требований, и нет необходимости проверять код при тестировании черного ящика. Это делается исключительно на основе точки зрения заказчика, только тестировщик знает набор входных данных и предсказуемые выходные данные.

1. Разделение эквивалентности: Этот метод делит входную область программы на классы эквивалентности, из которых могут быть получены тестовые примеры, поэтому он может уменьшить количество тестовых примеров.
2. Анализ граничных значений: он фокусируется на тестировании на границах или там, где выбираются экстремальные граничные значения. Он включает в себя минимальные, максимальные, только внутри / за пределами границ, значения ошибок и типичные значения.
3. Размытие: этот метод подает случайный ввод в приложение. Он используется для поиска ошибок реализации, используя ввод искаженных/полу-искаженных данных в автоматическом или полуавтоматическом сеансе.
4. Причинно-следственный график: В этом методе тестирование начинается с создания графика и установления связи между эффектом и его причинами.
5. Тестирование ортогонального массива: его можно применять там, где входная область очень мала, но слишком велика для проведения исчерпывающего тестирования.
6. Тестирование всех пар: В этом методе тестовые примеры предназначены для выполнения всех возможных дискретных комбинаций каждой пары входных параметров. Его основная цель - создать набор тестовых примеров, охватывающих все пары.

Рисунок 2. Представление различных форм тестирования черного ящика

7. Тестирование перехода состояния: Этот тип тестирования полезен для тестирования конечного продукта, а также для навигации по графическому пользовательскому интерфейсу.

Преимущества:

1. Тестировщикам не обязательно обладать знаниями конкретного языка программирования.
2. Тестирование проводится с точки зрения пользователя.
3. Это помогает выявить любые двусмысленности или несоответствия в спецификациях требований.
4. Программист и тестировщик независимы друг от друга.

Недостатки:

1. Тестовые примеры трудно разработать без четких спецификаций.
2. Вероятность повторения тестов, которые уже выполнены программистом.
3. Некоторые части серверной части вообще не тестируются.

B. Тестирование в белом ящике

Тестирование "белого ящика" в основном фокусируется на внутренней логике и структуре кода. Белый ящик выполняется, когда программист обладает полным знанием структуры программы. С помощью этого метода можно протестировать каждую ветвь и решение в программе.

1. Камеральная проверка: Камеральная проверка - это первичное тестирование кода. Авторы, которые очень хорошо владеют языком программирования, будут привлечены к кабинетному тестированию.
2. Пошаговое руководство по коду: В этом процессе тестирования группа технических специалистов просматривает код. Это один из видов полуофициального метода проверки.
3. Формальные проверки: Проверка - это формальный, эффективный и экономичный метод обнаружения ошибок в дизайне и коде.

4. Это официальная проверка, направленная на выявление всех неисправностей, нарушений и других побочных эффектов.
5. Тестирование потока управления: Это стратегия структурного тестирования, которая использует поток управления программой в качестве потока управления моделью и предпочитает больше, но более простых путей меньшему, но сложному пути.

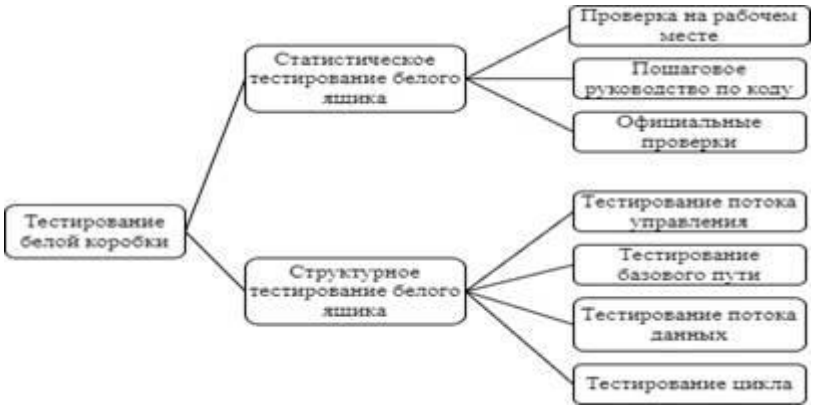

Рисунок 3. Представление различных форм тестирования белого ящика

5. Тестирование базового пути: тестирование базового пути позволяет разработчику тестового набора определить логическую сложность процедурного проектирования, а затем использовать эту меру в качестве подхода для определения базового набора путей выполнения.
6. Тестирование потока данных: В этом типе тестирования график потока управления снабжен комментариями с информацией о том, как определяются и используются переменные программы.
7. Тестирование цикла: оно фокусируется исключительно на валидности конструкции цикла.

Преимущества:

1. Он выявляет ошибку в скрытом коде, удаляя лишние строки кода.
2. Максимальный охват достигается во время написания тестового сценария[7].
3. Разработчик тщательно излагает причины реализации.

Недостатки:

1. Для проведения этого тестирования необходим опытный тестировщик, поскольку требуется знание внутренней структуры.
2. Многие пути останутся непроверенными, так как очень трудно заглянуть в каждый закоулок, чтобы обнаружить скрытые ошибки.

C. Тестирование серого ящика:

Тестирование "серого ящика" пытается объединить преимущества тестирования "черного ящика" и "белого ящика" и, как правило, успешно.

Тестирование "серого ящика" использует простой подход тестирования "черного ящика", но также использует некоторые ограниченные знания о внутренней работе приложения. Белый ящик + Черный ящик = Серый ящик, это метод тестирования приложения с ограниченными знаниями о внутренней работе приложения, а также со знанием фундаментальных аспектов системы. Таким образом, тестировщик может проверить как выходные данные пользовательского интерфейса, так и процесс, который приводит к этому выходу пользовательского интерфейса. Тестирование "серого ящика" может быть применено к большинству этапов тестирования; однако в основном оно используется при интеграционном тестировании.

1. Тестирование ортогонального массива: этот тип тестирования используется как подмножество всех возможных комбинаций.
2. Матричное тестирование: В матричном тестировании указывается отчет о состоянии проекта.
3. Регрессионное тестирование: Если в программное обеспечение вносятся новые изменения, регрессионное тестирование подразумевает запуск тестовых случаев.
4. Тестирование шаблонов: тестирование шаблонов проверяет хорошее приложение на предмет его архитектуры и дизайна.

Рисунок 4. Представление формы тестирования серого ящика

Преимущества:

1. Это обеспечивает комбинированное преимущество методов тестирования "черного ящика" и "белого ящика".
2. При тестировании серой коробки тестировщик может разработать отличные тестовые сценарии.
3. Беспристрастное тестирование
4. Создайте интеллектуальную систему разработки тестов.

Недостатки:

1. Охват тестированием ограничен, так как доступ к исходному коду недоступен.
2. Многие программные пути остаются непроверенными.
3. Тестовые примеры могут быть избыточными.

Таблица 1.
Сравнение трех форм методов тестирования

	Тестирование черного ящика	**Тестирование белого ящика**	**Тестирование серого ящика**
1.	Анализирует только фундаментальные аспекты, т.е. никаких знаний о внутренней работе.	Полное знание внутренней работы.	Частичное знание внутренней работы.
2.	Это наименее утомительно и отнимает много времени.	Потенциально наиболее исчерпывающий и отнимающий много времени	Что-то среднее между другими методами.
3.	Не подходит для тестирования алгоритмов.	Он подходит для тестирования алгоритмов (подходит для всех).	Не подходит для тестирования алгоритмов.
4.	Низкая степень детализации.	Высокая степень детализации.	Средняя степень детализации
5.	Выполняется конечными пользователями, а также тестировщиками и разработчиками (пользовательское приемочное тестирование).	Это выполняется разработчиками и тестировщиками.	Выполняется конечными пользователями, а также тестировщиками и разработчиками (пользовательское приемочное тестирование).

ВЫВОДЫ

Тестирование программного обеспечения - это деятельность, которая выполняет программное обеспечение с намерением найти в нем ошибки. Тестирование программного обеспечения может обеспечить независимое представление о программном обеспечении, позволяющее бизнесу оценить и понять риски, связанные с внедрением программного обеспечения. Чтобы более эффективно проводить тестирование программного обеспечения, в данной статье проводится сравнительное исследование трех основных методов тестирования программного обеспечения.