004.415.53

Лазченко В.Р.,
студент бакалавриата
4 курс, факультет «Информатика и вычислительная техника»
Научный руководитель: Остроух Е.Н., ст.пр.
Донской Государственный Технический Университет
Россия, г.Ростов-на-Дону

Lazchenko V.R.,
undergraduate student
4 course, faculty "Computer Science and Engineering"
***Scientific adviser: Ostroukh EN,* Senior Lecturer**
Don State Technical University
Russia, Rostov-on-Don

ВВЕДЕНИЕ В ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Аннотация

The article presents the concepts, classification, as well as some approaches at the stage of software testing.

Ключевые слова

Тестирование, цикл разработки, QA, IT.

INTRODUCTION TO TESTING SOFTWARE

Annotation

The article presents the concepts, classification, as well as some approaches at the stage of software testing.

Keywords

Testing, development cycle, QA, IT.

Процесс разработки программного обеспечения включает в себя ряд этапов. Одним из ключевых этапов, на котором определяется состояние программного продукта и принимаются решения о необходимости дальнейших изменений, является **тестирование**. Процесс тестирования включает в себя глубокий анализ текущих результатов в соответствии с техническим заданием, которое также предварительно анализируется на ряд критериев. Тестирование является отдельной дисциплиной с теоретической основной, собственными инструментами и отдельной командой исполнителей.

Теория тестирования включает в себя понятия, классификацию, принципы и подходы.

Тестирование программного обеспечения — проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом.

Также основными понятиями, связанные с процессом тестирования являются:

1. Quality Assurance (QA)
2. Quality Control (QC)
3. Testing

Quality Assurance (обеспечение качества) – совокупность мероприятий, включающая в себя все технологические аспекты разработки и внедрения программных систем с целью обеспечения необходимого уровня качества программного продукта.

Quality Control (контроль качества) – процесс контроля соответствия разрабатываемой системы к заранее регламентированным требованиям

Testing (тестирование) – процесс формирования и реализации тестовых сценариев.

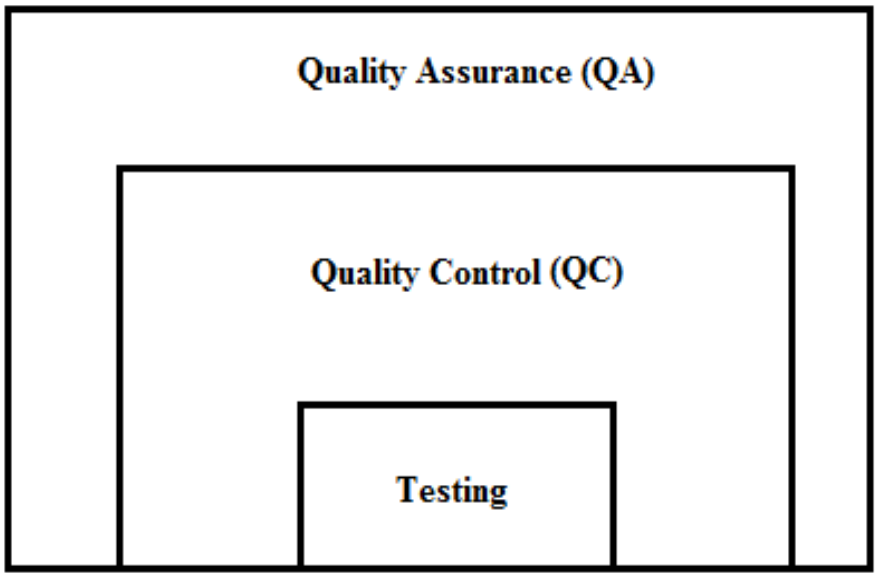

Схема процесса обеспечения качества

На схеме видно, что обеспечение качества включает в себя контроль качества и непосредственно само тестирование. Если с процессом тестирования более-менее понятно, то в чем же принципиальное отличие между обеспечением качества и контролем качества? Контроль качества (QC) контролирует процесс разработки продукта, когда обеспечение качества (QA) включает в себя все процессы, связанные с повышением качества программного продукта.

Цели тестирования:

1. Предоставление информации о состоянии разрабатываемого продукта
2. Повышение качества программного продукта

Повышение качества программного продукта включает в себя как локализацию и устранение дефектов, так и проверку на соответствие требованиям, указанным в технической документации.

Классификация видов тестирования

По объекту тестирования:

1. Функциональное тестирование
2. Тестирование пользовательского интерфейса

2.1 Тестирование UI

2.2 Тестирование локализации

3. Тестирование производительности

3.1 Нагрузочное

3.2 Стрессовое

3.3 Тестирование стабильности

3.4 Конфигурационное

4. Тестирование безопасности
5. Тестирование документации
6. Инсталляционное тестирование

По степени автоматизации:

1. Ручное
2. Автоматизированное

По знанию тестируемого объекта:

1. Тестирование методом «черного ящика»
2. Тестирование методом «серого ящика»
3. Тестирование методом «белого ящика»

По степени изолированности компонентов:

1. Модульное
2. Интеграционное
3. Системное

По характеру сценария:

1. Позитивное
2. Негативное

По моменту проведения:

1. Тестирование нового функционала
2. Регрессионное тестирование
3. Дымовое тестирование

В тестировании классификация относительно объемная, в некоторых моментах спорная и неоднозначная. Большая часть видов тестирования интуитивно понятна, исходя из названий. Однако есть и те, на которые стоит обратить особое внимание. Так, в классификации по знанию тестируемого объекта существует три метода. Первый – тестирование методом «черного ящика», обозначает процесс тестирования, при котором внутреннее устройство программного продукта (в частности программный код) неизвестен, анализ происходит исключительно, исходя из внешнего поведения программы. Тестирование методом «серого ящика» подразумевает под собой анализ поведения программы с частичным доступом к внутренностям программы. Примером использования этого метода служит запуск отладчика, который обеспечивает анализ программы во время её выполнения, предоставляя информацию о работе с памятью и взаимодействию с компонентами ОС. В случае, когда есть частичный доступ к исходному коду, метод также относится к тестированию «серого ящика». Метод «черного ящика» подразумевает полный доступ к исходному коду программы, за счет чего появляются существенные преимущества. При анализе кода можно обнаружить ошибки, которые обычными способами выявить проблематично. Зачастую это ошибки работы, связанные с памятью и взаимодействием с внешними данными. Также стоить обратить внимание и на регрессионное тестирование. Это вид тестирования, при котором анализируются ранее протестированные участки исходного кода с целью выявления ошибок после создания нового или изменения ранее созданного функционала. При регрессионном тестировании выявляется большая часть ошибок, что связано с нарастанием функционала и вероятности внутреннего взаимодействия, которое прогнозировать зачастую проблематично.

Этапы тестирования:

1. Анализ проекта
2. Разработки стратегии тестирования
3. Создание тестовой документации (тест-план, чек-листы, кейсы)

4. Тестирование прототипа
5. Основное тестирование
6. Стабилизация
7. Эксплуатация

Принципы тестирования:

1. Тестирование демонстрирует наличие дефектов, а не их отсутствие

Доказать полное отсутствие дефектов невозможно. Если помимо найденных не удается найти остальные дефекты, это не означает, что их нет.

2. Исчерпывающее тестирование недостижимо

В соответствии с первым принципом можно утверждать, что тестирование может быть лишь недостаточным, но никогда не избыточным. Невозможно провести тестирование, которое покрывало бы все комбинации данных и различного рода взаимодействий.

3. Принцип необходимости раннего тестирования

Необходимость раннего тестирования обусловлена увеличением проблематичности пропущенных на этапах возникновения ошибок. Высока вероятность, что по мере нарастания функционала, ошибку придется исправлять не только в месте первоначального её возникновения, но и учитывать её во всех местах использования модуля, в котором она возникла.

4. Скопление дефектов

В местах, где дефекты наиболее ожидаемы, требуется большее сосредоточение усилий при тестировании. Чаще этими местами являются модули, имеющие наибольшее количество связей с другими модулями, а также места использования изначально проблематичных компонентов.

5. Парадокс пестицида

Часто используемые тесты уменьшают кол-во ошибок в тех местах, которые эти тесты покрывают. При этом без внимания остаются остальные места, которые увеличиваются в процессе разработки.

6. Тестирование зависит от контекста

Приоритеты при тестировании определяются в соответствии с основной целью программного продукта. Например, хорошо посещаемый сайт информационного характера требует особого внимания к нагрузочному тестированию, когда сайт для ограниченного числа пользователей, связанный с базой данных, содержащую информацию конфиденциального характера, нуждается в тщательном тестировании безопасности.

Виды ошибок:

1. Error – ошибка пользователя при использовании программы непредусмотренным способом. Такие способы должен предусмотреть программист и назначить соответствующее уведомление об ошибке, иначе ошибка перейдет в разряд failure.
2. Bug – ошибка со стороны программиста, связанная с нарушенной логикой работы программы.
3. Failure – сбой в программе или её части, который может быть связан как с аппаратной несовместимостью, так и с внутренними проблемами используемых компонентов.

Уровни серьезности дефекта:

1. Блокирующий (S1) – уровень дефекта, при котором дальнейшая работа программы невозможна.
2. Критический (S2) – при данном дефекте программа не работает обычным способом, либо не работает основная часть функционала.
3. Значительный (S3) – дефект не блокирует основной функционал, но может препятствовать корректному выполнению как основному, так и дополнительному функционалу.

4. Незначительный (S4) – дефект, не препятствующий выполнению назначения программы.
5. Тривиальный (S5) – малозаметный дефект, не оказывающий влияния на работу программы.

Уровни приоритетов дефекта:

1. Высокий (P1) – дефект имеет высокий уровень значимости, должен быть исправлен в первую очередь.
2. Средний (P2) – дефект является существенным, его необходимо исправить после устранения более значимых дефектов.
3. Низкий (P3) – дефект имеет наименьшее значение, может быть исправлен в любое время.