Тестирование ПО

(продолжение)

Тестирование ПО

Разработка тестовых случаев

Тестовый сценарий (англ., Test Script) – это алгоритм проверки некоторой функциональности программы в совокупности с ожидаемыми результатами. Примером тестового сценария может быть,например, проверка функциональности добавления некоторой сущности в базу данных. В процессе добавления возможны разные случаи:

Тестирование ПО

верно заполнить все поля сущности, выборочно верно заполнить поля, неверно заполнить поля, оставить одно поле пустым, оставить два поля пустыми и т. д. Все отдельные случаи проверок называются тестовыми случаями (англ., Test Case). Тестовые случаи организуются в тестовые наборы (англ., Test Suite), например, режим работы администратора или незарегистрированного пользователя. Тестовые случаи могут изменяться в течение работы над проектом и полностью формируются по мере понимания задачи, по мере изменения требований, а также по другим причинам. Процесс разработки тестовых случаев состоит из следующих этапов:

Тестирование ПО

1. Ознакомление с проектной документацией: требованиями к программному продукту (либо спецификацией), тестовым планом и непроектной информацией (например, литературой по предметной области для разрабатываемого ПО).
2. Проектирование тестовых случаев. При разработке тестовых случаев актуальным понятием является модульность. При модульном проектировании система разбивается на отдельные компоненты, каждый из которых имеет свое назначение и четко определенные входы и выходы. В функциональном тестировании этим модульным компонентом является тестовый случай.

Тестирование ПО

Это значит, что перед каждым тестовым случаем должна быть поставлена четко сформулированная цель (что именно подвергается тестированию), определена среда тестирования с известными начальными условиями (благодаря чему можно рассчитывать, что при каждом прогоне конкретный тест будет выдавать один и тот же результат). Также один тестовый случай должен проверять только одну функциональность и иметь строго определенный ожидаемый результат (чтобы было понятно успешно или неуспешно прошло испытание).

Формирование тестовых наборов связано с градацией тестов, из которой наиболее распространенной является разбивка на три блока:

Тестирование ПО

– тесты для проверки «на дым» (англ., Smoke Test) – это краткая и быстрая проверка основной функциональности программы с целью принять или отклонить версию программы для дальнейшего тестирования;

– тесты для позитивного тестирования – проверка работы программы в стандартных ситуациях (вводятся верные данные, нажимаются нужные кнопки, имитируется работа «положительного» пользователя);

– тесты для негативного тестирования – проверка работы программы в нестандартных ситуациях (вводятся неверные данные, нажимаются ненужные кнопки, имитируется работа «плохого»

Тестирование ПО

пользователя.

3. Написание тестовых случаев. На этом этапе необходимо придерживаться следующих свойств тестовых случаев:

a) тестовый случай должен обладать высокой вероятностью обнаружения дефекта (для этого тестировщик должен стать на позицию конструктивного разрушения);

b) тестовый случай должен быть воспроизводимым;

c) тестовый случай должен обладать четко определенными ожидаемыми результатами и критериями успешного выполнения теста;

Тестирование ПО

d) набор тестовых случаев не должен быть избыточным.

4. Проверка тестовых случаев. По окончании разработки тестовых случаев проводится их проверка, в течение которой обращают внимание на следующие моменты:

– соответствует ли тестовый случай функциональности, заявленной в требованиях;

– покрывают ли тестовые случаи все требования к программному продукту (для этого используется матрица прослеживаемости требований);

– организованы ли тестовые случаи достаточно эффективно, чтобы можно было обходиться

Тестирование ПО

минимальными конфигурациями средств тестирования;

– включены ли тесты в систему управления конфигурациями;

– имеются ли среди тестов избыточные, можно ли устранить эту избыточность;

– снабжен ли каждый тестовый случай ожидаемыми результатами;

– достаточно ли подробно разработана методика тестирования, чтобы стала возможной ее автоматизация.

Тестирование ПО

Выбор тестовых данных

Для сокращения переборов область всех входных данных программы можно разбить на конечное число классов эквивалентности. Такое разбиение основано на принципе эквивалентности из общей теории систем, который в данном контексте можно выразить следующим образом: если система реагирует некоторым образом на входное значение x1, то существует довольно большая вероятность, что система отреагирует аналогичным образом на входное значение x2, очень близкое к x1. Тогда все множество входных данных системы можно разбить, как минимум, на два класса эквивалентности: класс верных значений (они войдут в позитивный тест) и

Тестирование ПО

класс неверных значений (для негативного теста).

Граничными значениями будем называть те значения, которые располагаются на границах классов эквивалентности, а эквивалентными – те, которые расположены внутри классов эквивалентности.

Эквивалентные значения из класса неверных могут быть абсолютно разными, но выбранными с целью поломки программы и проверки ее реакции в непредвиденных ситуациях. Независимо от классов эквивалентности рекомендуется проверять 0, –1, +1 и специальные символы.

Тестирование ПО

Функциональное тестирование

Ошибка, свойства ошибки

Функциональное тестирование направлено на поиск ошибок в заявленной функциональности программы. Поэтому ключевым понятием здесь является термин «ошибка». Существует довольно много определений ошибки, приведем некоторые из них:

– ошибка – это расхождение между программой и ее спецификацией. Определение часто критикуют, поскольку пользователь программы может не иметь спецификации, но на экране увидеть, например, Error 404.

Тестирование ПО

– если программа не делает того, чего пользователь от нее вполне обоснованно ожидает, значит налицо программная ошибка;

– не существует ни абсолютного определения ошибок, ни точного критерия наличия их в программе, можно лишь сказать, насколько программа не справляется со своей задачей – это исключительно субъективная характеристика.

Последние два определения имеют право на существование, однако для краткости будем называть ошибкой несоответствие ожидаемых результатов с фактическими полученными.Выделим следующие свойства ошибок:

Тестирование ПО

1) Важность. В этом свойстве могут быть определены следующие уровни:

– критическая важность, когда произошел сбой либо отказ системы, и дальнейшая работа с программой невозможна;

– серьезная важность, когда не работает основная функциональность программы (например, добавление записи в базу данных);

– средняя важность, когда не работает второстепенная функциональность либо имеются недочеты в работе основной функциональности (например, не работает сортировка товара в списке, или не появляется сообщение для подтверждения удаления товара из базы данных);

Тестирование ПО

– низкая важность, когда имеется мелкая ошибка (например, ошибка интерфейса либо орфографическая ошибка).

2) Воспроизводимость. Это свойство связано с частотой и условиями воспроизведения, поэтому здесь выделяют два уровня:

– всегда, т. е. ошибка воспроизводится на всех устройствах и не зависит ни от каких условий;

– иногда, т. е. ошибка воспроизводится только при определенных условиях, например, только в определенном браузере.

Тестирование ПО

3) Приоритет. Это свойство связано со скоростью исправления ошибки. Можно выделить следующие уровни приоритета:

– очень высокий, когда ошибка должна быть исправлена немедленно;

– высокий, когда ошибка должна быть исправлена как можно скорее;

– средний, когда ошибка может быть исправлена, когда появится свободное время;

– низкий, когда ошибка может быть исправлена в последнюю очередь.

Тестирование ПО

4) Симптом. Это свойство также называют категорией ошибки. К одной ошибке иногда может подходить несколько симптомов, поэтому при документировании ошибки достаточно указать один из них, но, желательно, самый точный. Перечень симптомов может отличаться в зависимости от используемой системы управления ошибками, поэтому приведем наиболее часто встречающиеся:

– неожиданное поведение, когда, например, вместо сообщения о сохранении данных, появляется сообщение об их успешном удалении;

– недружественное поведение, когда, как правило, программа не сопровождает свои действия сообщениями и предупреждениями;

Тестирование ПО

– неверное действие, когда программа не выполняет требуемое действие или выполняет неверно, например, не удаляет запись из базы данных;

– отказ системы, когда дальнейшая работа с программой невозможна;

– потеря данных, когда, например, добавленные данные исчезли;

– искажение данных, когда произошла замена данных программы искаженными, или, например, данные выводятся на экран в другой кодировке;

– низкая производительность, когда, например, слишком медленно идут вычисления, или медленно раскрывается таблица;

Тестирование ПО

– ошибка локализации может быть связана с воспроизведением на разных устройствах, а также, например, в связи со сменой локализации сайта на другой иностранный язык;

– инсталляционная ошибка, которая проявляется во время разных способов инсталляции ПО;

– косметическая ошибка связана с интерфейсом программы;

– ошибка документации связана с недочетами в любой документации, используемой в процессе разработки ПО;

Тестирование ПО

– различия со спецификацией, когда, например, названия элементов интерфейса, их количество, цвет, внешний вид и т. д. не соответствуют заявленным в требованиях или в спецификации к программному продукту;

– отсутствующая функциональность, когда функциональность, заявленная в требованиях, отсутствует в программном продукте (например, отсутствует поле для поиска и, соответственно, вся функциональность поиска);

– запрос на улучшение является симптомом, который может быть выставлен, когда тестировщик предлагает улучшить некоторую функциональность программы либо ее внешний вид.

Тестирование ПО

Таким образом, ошибка с этим симптомом может не противоречить требованиям, и, соответственно, может не быть исправлена.

Тестирование ПО

Составления отчетов об ошибках

При тестировании обычно заполняется таблица со следующими полями:

- № теста;
- № требования;
- Название модуля/экрана;
- Описание тестового случая;
- Ожидаемые результаты;
- Тест пройден? Да/Нет.

Тестирование ПО

Процесс поиска ошибок, как правило, состоит в прохождении одного за другим заранее запланированных тестовых случаев, сверки ожидаемых результатов с фактически полученными и вывода, прошел тест или нет. При этом заполняется последняя колонка приведенной таблицы тестирования.

Если тест не пройден, значит найдена ошибка, которая должна быть документирована и передана программисту на исправление. Для того чтобы исправление ошибки проходило без особых затруднений, отчет об ошибке должен составляться быстро, но при этом его тон и содержание должны максимально способствовать решению проблемы.

Тестирование ПО

Необходимо проанализировать ошибку, воспроизвести ее несколько раз, выяснить условия воспроизведения, данные, при которых она появляется.

И только потом описать ее предельно кратко и четко, поскольку слишком пространное и расплывчатое описание проблемы затрудняет понимание ее сути.

Кроме свойств ошибки, описанных выше, необходимо заполнить поля, приведенные далее в таблице 14.1.

Тестирование ПО

Таблица 14.1 - Информация об обнаруженной ошибке

Характеристика	Описание	Обоснование
Идентификатор ошибки	Уникальный идентификатор, обычно строка из алфавитно-цифровых символов.	Позволяет отслеживать ошибку в процессе изменения ее состояний.
Статус ошибки	Один из допустимых статусов.	Присвоить ошибке статус «Новый» в момент обнаружения.
Кем обнаружен	Имя исполнителя, обнаружившего дефект.	Обеспечивает возможность задавать вопросы относительно дефекта.
Дата обнаружения	День/месяц/год.	Позволяет отслеживать хронологию событий, связанных с ошибкой.

Тестирование ПО

Таблица 14.1 - Информация об обнаруженной ошибке (продолжение)

Краткое описание проблемы	Описание на концептуальной форме, как правило, одной строкой.	Описание используется в сообщениях о статусе ошибки.
Описание проблемы	Детальное описание симптомов проблемы и того, как она ограничивает функциональные возможности продукта.	Предназначено для тех, кому необходимо детальное описание проблемы, например, разработчику, который работает над устранением ошибки.
Как воспроизвести проблему	Детально проработанная методика воспроизведения проблемы.	Позволяет разработчикам локализовать проблему.

Тестирование ПО

Ошибки, как правило, сохраняют в автоматизированные системы документирования и отслеживания ошибок. Каждая из таких систем имеет свои особенности при составлении отчетов об ошибках и поля для заполнения, однако можно предложить следующие общие правила:

1) Составляйте отчет об ошибке сразу же после ее обнаружения, иначе про нее можно забыть.
2) Не составляйте отчет на бумаге (поскольку листок может легко потеряться), а заносите в систему документирования и отслеживания ошибок.
3) Составьте полное описание ошибки с указанием операционной системы, браузера, базы данных и/или других подробностей.

Тестирование ПО

4) Опишите подробно шаги для воспроизведения ошибки, чтобы разработчик мог их повторить и увидеть ошибку.
5) Придумайте краткое, но емкое название для ошибки.
6) Не путайте описание ошибки и шаги воспроизведения, а также название ошибки и описание ошибки.
7) Укажите важность ошибки, симптом, частоту появления и приоритет.
8) Пользуйтесь простым и доходчивым языком при составлении отчета об ошибке.

Тестирование ПО

9) Не обвиняйте никого в обнаруженной ошибке, поскольку отчет не должен приводить к конфликту между тестировщиками и программистами, а лишь способствовать разработке качественного программного продукта.

Системы документирования/отслеживания ошибок

Системы документирования и отслеживания ошибок (Bug Tracking Systems, BTS) в большинстве своем, не ограничиваются работой с ошибками, а представляют собой системы управления проектами. Они позволяют отслеживать процесс тестирования, проверять и составлять отчеты.

Тестирование ПО

Общее назначение таких систем можно описать следующим образом:

1) повышать взаимодействие между сотрудниками;
2) ни одна ошибка не должна быть не исправлена, потому что так решил разработчик;
3) как можно меньше ошибок должно остаться из-за проблем взаимодействия сотрудников.

Несмотря на то, что существует довольно большой перечень BTS, все их количество можно разбить на классы:

– свободно распространяемые (например, Bugzilla, Mantis, Redmine, Track);

Тестирование ПО

– платные системы (например, Jira, ClearQuest, TestTrackPro);

– собственная разработка (т. е. компания разработала для себя такую систему).

Следует отметить, что выбор BTS зачастую определяется заказчиком программного обеспечения.

Молодые и немногочисленные компании останавливаются на бесплатных системах, а крупные компании в состоянии позволить себе собственную разработку по своим требованиям.

Можно предложить следующие критерии выбора BTS:

Тестирование ПО

1) доступная система для различных платформ;
2) поддержка работы с различными базами данных;
3) стоимость и схема лицензирования;
4) интегрирование в различные среды;
5) гибкость настройки системы;
6) электронная поддержка формирования событий и отчетов.

Жизненный цикл ошибки

Жизненный цикл ошибки – это путь, который проходит ошибка с момента ее обнаружения до закрытия либо другого терминального статуса. Каждый этап жизни ошибки характеризуется ее статусом.

Тестирование ПО

Не все BTS имеют одинаковые названия статусов ошибок и, соответственно, жизненные циклы. Рассмотрим основные из них (рис. 14.5).

Жизненный цикл ошибки начинается с ее обнаружения и присвоения статуса *новая* (или *найдена*, или *обнаружена*, или *submitted*). Затем ошибка направляется разработчику для исправления и получает статус *исправить* (или *назначить*, или *assigned*). После исправления ошибка получает статус *исправлена* (или *fixed*). Затем исправленная ошибка проверяется тестировщиком еще раз (как правило в новой версии ПО) и, если она не воспроизводится, то получает статус *проверена* (или *закрыта*, или *verified*).

Тестирование ПО

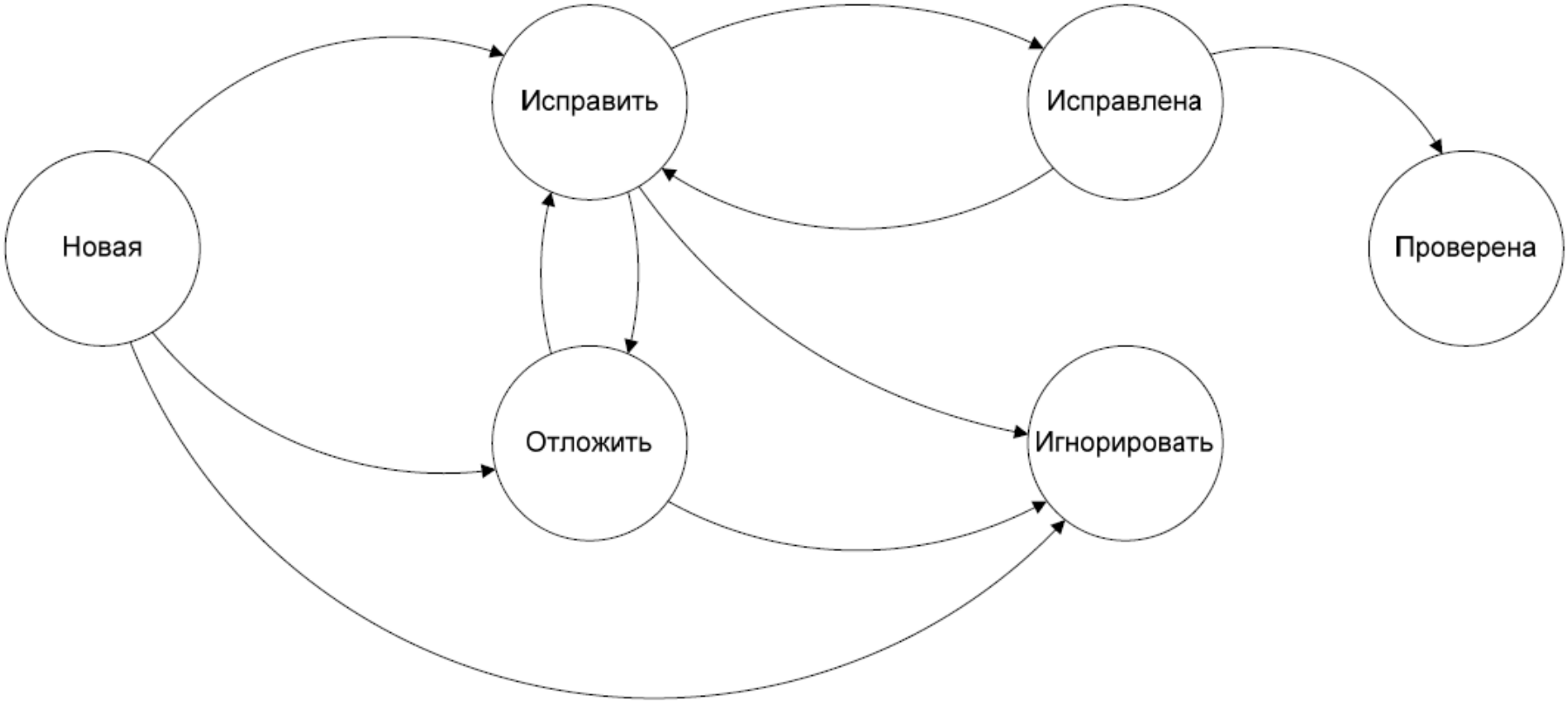

Рис. 14.5 - Жизненный цикл ошибки

Тестирование ПО

На этом жизненный цикл ошибки заканчивается, однако это идеальный случай, который не всегда происходит. Возможны и другие пути жизненного цикла ошибки:

1. Исправленная ошибка повторяется снова, тогда она отправляется на рассмотрение по новому кругу.
2. Документированная ошибка не воспроизводится, тогда ей может быть присвоен статус *игнорировать* (или *не воспроизводится*, или *declined*).
3. Возможны также случаи, когда документированная ошибка, например, не может быть исправлена в текущей версии, тогда ее откладывают с присвоением статуса *отложена* (или *deffered*).

Тестирование ПО

Возможны и другие статусы ошибок, определяемые системами документирования ошибок, например, статус дубликат, когда в BTS уже документирована такая ошибка, или статус не ошибка, когда тестировщик ошибочно составил отчет.

Реализация тестов

Приемочный тест

Как указывалось ранее, тест на «дым» или приемочный тест – это быстрая проверка выполнения основной функциональности программы. После приемочного теста делается вывод о пригодности программы для дальнейшего тестирования или нет.

Тестирование ПО

Каждая версия разрабатываемого ПО подвергается проверке «на дым», поскольку нет смысла тратить усилия тестировщиков на версию, в которой не работает даже основная функциональность.

Критерии не прохождения приемочного теста часто зависят от специфики решаемой задачи и должны быть отражены в тестовом плане, однако можно выделить общие из них:

1) отсутствие каких-либо файлов или компонент, без которых любое тестирование ПО невозможно;
2) сбой программного обеспечения или системы в начале работы, и дальнейшее тестирование невозможно;

Тестирование ПО

3) ошибка в программе, приводящая к сбою в середине работы и делающая дальнейшее тестирование невозможным;

4) достижение определенного процента ошибок, недопустимого для приемочного тестирования, например, 20–25 % тестовых случаев не проходит (как правило, недопустимый процент указывается в тестовом плане).

Позитивный тест

Позитивный тест – это основной вид функционального тестирования, при котором проверяется типичная, логически верная работа программы.

Тестирование ПО

Этот вид тестирования проводят в случае пройденного приемочного теста посредством запуска программы и проверки работы каждой функциональности по заранее подготовленным тестовым случаям. Отработанный тестовый случай помечается как «пройденный» или «не пройденный». Для этого в таблице с тестовыми случаями существует отдельная колонка.

Если тест не пройден, тестировщик составляет отчет об ошибке и сохраняет его в BTS. Если в процессе тестирования обнаруживается ошибка, не предусмотренная тестовыми случаями, то тестировщик также составляет отчет об ошибке и добавляет новый тестовый случай в список.

Тестирование ПО

Такой подход позволит не потерять возможную ошибку в следующих версиях тестируемой программы.

Негативный тест

Негативный тест проверяет работу программы в непредвиденных и нестандартных ситуациях, например, при некорректно вводимом значении, при незаполненных полях для ввода, при дублировании данных и т. д. Негативный тест часто объединяют с мелкими проверками (например, на разный формат даты или перемещение по элементам интерфейса через табуляцию), тогда он получает название «углубленный тест» (или расширенный, англ., Extended Test).

Тестирование ПО

Такой тест проводят на стабильно работающей программе ближе к окончанию разработки, что связано с экономией времени, поскольку нет смысла проверять нестандартную работу программы, если не работают стандартные ситуации. Негативный тест проводят на основе заранее разработанных тестовых случаев, а также с использованием контрольных перечней (листов проверок, чек-листов, Check List).

Чек-листы – это мини-тестовые случаи, которые создаются для стандартных элементов и полей приложений, применяемых практически в любых проектах, например, тестовых полей, числовых, полей даты, времени и т. д.

Тестирование ПО

Поэтому нет смысла для каждого отдельного проекта разрабатывать свои тестовые случаи для таких полей и элементов, а достаточно использовать контрольные перечни, выработанные опытным путем в течение времени каждым тестировщиком, и имеющиеся, как правило, в каждой компании. Рассмотрим некоторые из них.

Проверка одного текстового поля:

– проверить на заполнение (должно быть первоначально пустым или нет);

– проверить на пробелы в начале и в конце текста (если не оговорено в требованиях, то пробелы должны быть опущены);

Тестирование ПО

– проверить на одни пробелы в поле;
– проверить на пробелы в середине текстового поля;
– проверить на специальные символы;
– проверить на символы конкретного языка, если имеется реализация приложения на нескольких иностранных языках;
– проверить на минимальную и максимальную длину поля;
– проверить на работу с кнопками <Shift>, <Insert>, <CapsLock>, а также на выделение текста.

Тестирование ПО

Проверка нескольких текстовых полей:

– проверить по всем пунктам чек-листа для одного текстового поля, а также на сохранение текста, состоящего только из одной строки.

Проверка числовых полей:

– проверить на минимально и максимально допустимое число;

– проверить на отрицательные значения;

– проверить на буквенные символы;

– проверить на 0, 1, –1.

Тестирование ПО

Проверка ListBox:

– проверить на выпадение списка по стрелке и комбинации клавиш <Ctrl> + <A>;

– проверить сортировку в списке (если не оговорено иначе, то должна быть по алфавиту);

– выделенное значение отображается и при сохранении не должно сброситься.

Проверка ComboBox:

– проверить внешний вид бокса;

– проверить на ввод своих значений – они должны отображаться.

Тестирование ПО

Проверка CheckBox:

– проверить работу с помощью мыши и клавиатуры.

Проверка RadioButton:

– проверить на установку позиции по умолчанию;

– проверить выделение поля после его выбора.

Поле валюты:

– проверить на количество знаков для разменной монеты;

– проверить точку или запятую для обозначения целой части.

Поле даты:

– проверить на пустую дату;

Тестирование ПО

– проверить на минимальную и максимальную дату, допустимые для этого поля;

– проверить, чтобы минимальная дата не была больше максимальной (для случая работы с периодом времени);

– проверить на некорректную дату;

– проверить на формат даты (программа должна реагировать на неформатную дату или уметь ее конвертировать). Возможны форматы дат: дд.мм.гггг (Беларусь, Россия, Германия), дд-мм-гггг (Италия, Дания, Нидерланды), дд/мм/гггг (Великобритания, Латинская Америка), мм-дд-гггг (США), мм/дд/гггг (Литва), гггг-мм-дд (Канада, Бельгия, Швеция) и др.

Тестирование ПО

Кнопки:

– проверить расположение кнопки и ее вид;

– проверить текст на кнопке (нельзя использовать аббревиатуру и сокращения);

– проверить, поместится ли текст на кнопке при переходе на другой иностранный язык;

– проверить на однократное и двойное нажатие на кнопку, а также на табуляцию;

– проверить доступность кнопки согласно требованиям.

Окна:

– проверить на позиционирование окна в зависимости от разрешения экрана;

Тестирование ПО

– проверить цветовую гамму;

– проверить появление скроллинга в необходимых случаях;

– проверить, чтобы все текстовые поля были выровнены, а шрифты были одинаковые;

– проверить, верный ли номер версии стоит в окне About.

Особенности тестирования

Все приложения принято делить на две группы: веб-приложения и standalone-приложения. Веб-приложения классифицируются по размеру проекта, по сфере использования и по целевому устройству.

Тестирование ПО

Standalone-приложения – по архитектуре, по принципу хранения данных, по рабочей платформе и по функционированию.

Общими видами тестирования для всех приложений являются:

1) Функциональное тестирование.
2) Тестирование интерфейса.
3) Тестирование удобства использования.
4) Тестирование производительности (для standalone-приложений учитывают время старта и запуска, время загрузки формы и отчета, для веб-приложений – время отклика страниц).

Тестирование ПО

5) Тестирование безопасности (для standalone-приложений проверяется корректность работы с операционной системой, лицензирование, устойчивость к случайным сбоям и проверка разграничений прав доступа, для веб-приложений – зависимость от браузера, устойчивость к инъекциям, модификации get и post запросов, изменение информации в URL, прямые запросы в базу, проверка на уязвимость, проверка разграничений прав доступа).
6) Тестирование локализации – это проверка на соответствие локальным стандартам, языковым стандартам и пользовательского интерфейса.

Тестирование ПО

7) Тестирование доступности (включая людей с ограниченными возможностями), реагирование на CAPTCHA, отключение изображений для экономии трафика.
8) Тестирование интеграции с другими приложениями.

Особенными видами тестирования для standalone-приложений являются следующие:

a) кросс-платформенное тестирование. Приложение не должно быть зависимо от операционной системы, если иное не написано в требованиях;
b) тестирование использования ресурсов (утечка памяти);

Тестирование ПО

c) тестирование установки программ и лицензирования. Используются различные способы установки, обращается внимание на типы установок, интерфейс установщика, роли пользователя, язык установки, сообщения в процессе установки, доступность приложения в процессе установки;

d) тестирование механизма обновления;

e) деинсталляция и проверка после этого состояния системы;

f) проверка работы программы на заданной конфигурации и запуск всех поддерживаемых локализаций.

Тестирование ПО

К специальным видам тестирования веб-приложений можно отнести следующие:

a) кросс-браузерное тестирование, т. е. проверка работы приложения в браузерах, предусмотренных в документе требований к программному продукту;

b) тестирование для мобильных устройств.

Должны быть специальные версии для этих устройств или проверка на разрешение экрана. Как правило, для такого тестирования используются эмуляторы;

c) поиск поврежденных ссылок и отсутствующих страниц.

Тестирование ПО

Автоматизация функционального тестирования

Автоматизация тестирования – это процесс замены ручного тестирования некоторым инструментальным средством.

Как правило, этот процесс состоит из следующих этапов:

1) принятие решения об автоматизации тестирования;
2) выбор инструментальных средств тестирования;
3) планирование и проектирование тестирования;
4) выполнение и управление тестированием;
5) оценка результатов тестирования.

Тестирование ПО

Наиболее распространенной формой автоматизации функционального тестирования является тестирование приложений через графический пользовательский интерфейс (GUI). Популярность такого вида тестирования объясняется двумя факторами: во-первых, приложение тестируется тем же способом, которым его будет использовать человек, во-вторых, можно тестировать приложение, не имея при этом доступа к исходному коду.

На сегодняшний день для автоматизации функционального тестирования через GUI на рынке представлено множество инструментов, среди которых можно выделить SilkTest, Win Runner,

Тестирование ПО

Rational Robot, Test Complete, QTP, Watir, Sahi, MS Coded UI и семейство Selenium.

Каждый из приведенных инструментов обладает определенными преимуществами и недостатками, а также спецификой применения.

Например, TestComplete поддерживает различные виды автоматизации тестирования, включая модульное, функциональное и нагрузочное, однако является очень дорогим.

Но для учебных целей существует возможность скачать демонстрационную версию на 30 дней с официального сайта производителя.

Тестирование ПО

Семейство Selenium, наоборот, является бесплатным, но позволяет тестировать только веб-приложения.

Общим свойством всех инструментов является получение тестового автоматизированного скрипта.

Если рассматривать области применения автоматизации, то наибольший эффект ее применения наблюдается с тестами, которые необходимо проводить большое количество раз, на разных наборах операционных систем и браузеров, при работе с большим количеством данных и для имитации большого количества пользователей, при длинных сценариях, с труднодоступными местами в системе (back-end процессы, работа с лог-файлами,

Тестирование ПО

запись в базу данных), при проверке данных, требующих точных математических расчетов.

Также выгодно использовать автоматизацию для регрессионного тестирования (набора тестов, которые нужно повторять снова и снова, чтобы быть уверенным, что приложение работает корректно, несмотря на вносимые изменения), для приемочного и углубленного тестов, чтобы проверять основную функциональность работы программы, и объемные негативные проверки.

Не нужно забывать, что автоматизацию следует применять только на стабильно работающем программном продукте.

Тестирование ПО

Плюсы/минусы автоматизации тестирования

Среди достоинств автоматизации функционального тестирования рассмотрим следующие:

1. Автоматизированные тесты могут прогоняться множество раз.
2. Большая скорость выполнения теста по сравнению с ручным.
3. Протекают каждый раз одинаково.
4. Повышается качество регрессионного тестирования.
5. Повышается качество проверки на совместимость.
6. Возможна автоматизация однообразных и однотипных задач.

Тестирование ПО

7. Возможность прогона тестов ночью и в выходные дни.
8. Автоматизированные испытания генерируют отчеты и журналы о выполнении.
9. Уменьшаются временные затраты на формирование статистических данных о проекте.
10. Возможно проведение нагрузочного тестирования, а также тестирование под разными серверами для каждой отдельной конфигурации и на разных иностранных языках.
11. Исключение ошибок, связанных с человеческим фактором.

Тестирование ПО

12. Управление несколькими машинами одновременно (распределенный тест).
13. Перезагрузка машины в другую операционную систему и вход под разными пользователями.
14. Взаимодействие с базами данных напрямую.
15. Поиск поломанных ссылок на веб-страницах.
16. Создание теста на одном браузере, а запуск его на другом.
17. Освобождение человека от рутинной работы с возможностью заняться более интересными и креативными задачами.

Тестирование ПО

Недостатков у автоматизированного тестирования не так уж и много. Обычно принято указывать, что разработка и отладка тестов требует больших затрат времени и средств, а также, что автоматизированные тесты не находят новых ошибок (только те, что тестировщик может сам предусмотреть, ведь, выполняя тест вручную, человек может обратить внимание на некоторые детали и, проведя несколько дополнительных операций, найти ошибку, а автоматизированный скрипт этого сделать не может).

К недостаткам можно еще отнести и стоимость инструмента для автоматизации: в случае, если используется лицензионное ПО.

Тестирование ПО

Большой список достоинств и мелкий недостатков может произвести впечатление идеального подхода к тестированию через ее автоматизацию. Развеять этот миф поможет перечень необоснованных ожиданий от автоматизации тестирования:

1. Автоматизировать можно все что угодно (ложь, поскольку нельзя автоматизировать тесты, где нельзя обойтись без участия человека, например, тестирование удобства использования).
2. Можно обнаружить бóльшее количество ошибок (ложь, поскольку автоматизированные тесты работают тоже по тестовым случаям, разработанным тестировщиком).

Тестирование ПО

3. Можно исключить или значительно сократить ручное тестирование (следствие из п. 1).
4. Возможно 100 % покрытие функциональности (следствие из п. 1).
5. Все необходимое тестирование может выполнять одно инструментальное средство (средства автоматизации имеют свою специфику, поэтому одним инструментом все покрыть не получится).
6. Временной график тестирования сократится (по статистике, на разработку одного автоматизированного тестового случая уходит до 15 раз больше времени, чем на прохождение его вручную).

Тестирование ПО

7. Автоматизация недорога (для проведения автоматизации тестирования необходимы квалифицированные специалисты с немалой заработной платой, а также большинство инструментов являются платными).
8. Средства автоматизации просты в использовании (во-первых, чтобы научиться работе с этими инструментами необходимо пройти специальное обучение, а во-вторых, инструменты автоматизации довольно капризны в использовании, и, зачастую, причины не прохождения тестов очень трудно определить).

Тестирование ПО

Требования к автоматизированным тестам

Для того чтобы начинать разрабатывать автоматизированные тесты, необходимо определиться со следующими вопросами:

– Какую функциональность нужно автоматизировать?
– Какие действия необходимы для проведения теста?
– Где брать тестовые данные?
– Каков ожидаемый результат теста?
– Каковы критерии успешного прохождения теста?

Выделим основные требования к автоматизированным тестам:

Тестирование ПО

1) Тест должен тестировать сам, т. е. не только эмулировать действия пользователя с программой, но и проводить проверки.
2) Тест должен выдавать результат в таком виде, чтобы сразу можно было понять, где ошибка.
3) Один тест должен проверять только одну функциональность либо один случай использования функциональности.
4) Каждый тест должен быть независимым (т. е. выполняться независимо от других тестов).
5) Тест должен запускаться на разных платформах и в разных браузерах.

Тестирование ПО

6) Наличие документированных требований или пояснений к запуску теста.
7) Тест должен одинаково хорошо работать на быстрой машине и медленной.

Методы автоматизации ф-го тестирования

Метод Play&Record

Метод Play&Record (в литературе также встречается название Record&Playback) – это возможность работать в режиме записи действий, совершаемых над приложением, и повторное воспроизведение их автоматически.

Тестирование ПО

Записанные действия преобразуются в программный код на языке программирования, который встроен в инструмент автоматизации.

Последовательность шагов в этом методе такова:

1. Из инструмента автоматизации происходит захват тестируемого приложения.
2. Включается запись действий.
3. Выполняются требуемые действия над программой.
4. Останавливается запись.
5. Включается проигрывание записанных действий.

Тестирование ПО

Записанные действия отображаются в виде программного кода на встроенных языках программирования, в качестве которых могут быть как собственные языки инструментов, например, 4Test для инструмента SilkTest, так и распространенные языки программирования, например, C# или JavaScript в TestComplete.

В полученный таким образом скрипт можно вносить изменения, проверки, циклы, формировать логику теста и т. д.

К недостаткам метода Play&Record можно отнести следующие:

Тестирование ПО

1) Скрипты, получаемые этим методом, содержат фиксированные значения, которые должны быть изменены каждый раз, когда в приложении что-то меняется.
2) Стоимость, связанная с поддержкой автоматизированных скриптов, полученных по этому методу, астрономическая и неприемлемая, поскольку их приходится часто переписывать.
3) Полученные скрипты не надежны и часто не работают, даже если тестируемое приложение не изменялось.

   Причинами здесь могут быть различные всплывающие окна, сообщения или другие события,

Тестирование ПО

которые происходили либо не происходили в момент записи теста, и, соответственно, которые будет ожидать скрипт во время повторного воспроизведения либо неожиданно произойдут для скрипта.

4) Если тестировщик совершает ошибку, например, во время ввода данных, тест должен быть записан заново.