Manual QA

Lecture 7. Виды тестирования.

Виды тестирования

1) По знанию внутренностей системы
2) По объекту тестирования
3) По субъекту тестирования
4) По времени проведения тестирования
5) По критерию “позитивности” сценариев
6) По степени автоматизированности тестирования
7) По степени подготовленности к тестированию

1. По знанию внутренностей системы

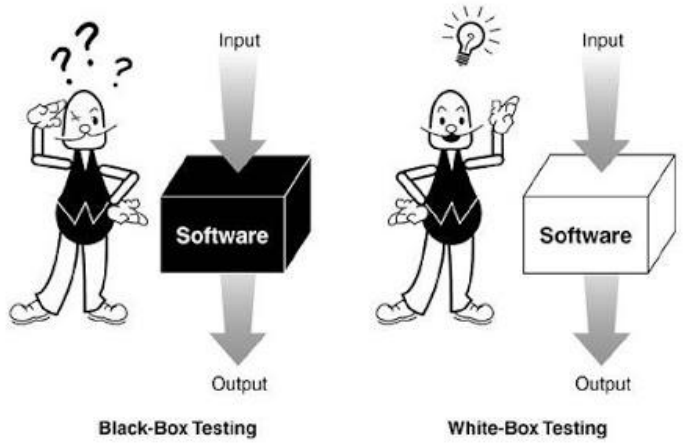
- Тестирование черного ящика (Black Box Testing)
- Тестирование серого ящика (Grey Box Testing)
- Тестирование белого ящика (White Box Testing)

Тестирование черного ящика

- Black Box (closed box, opaque box, behavioral) Testing

- Тестировщик производит тестирование не имея информации о том, как устроена система изнутри.

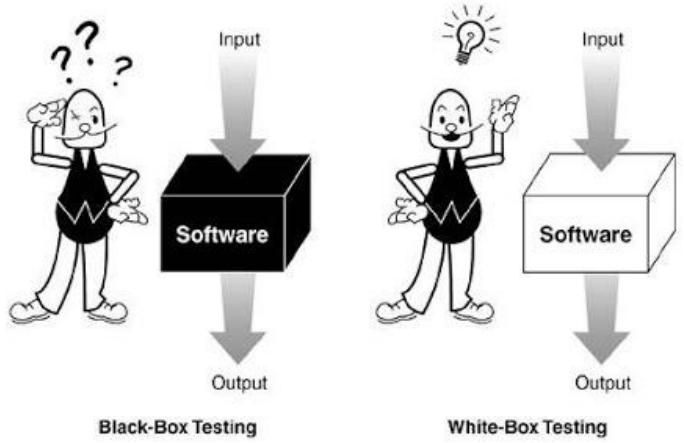
- Идеи для тестирование идут от предполагаемого поведения пользователей.



Тестирование белого ящика

- White (Glass box, structural) Box Testing

- Противоположность методу Черного ящика
- Тестирование производится на основании информации, как устроена система изнутри
- Обычно производится самими программистами



Тестирование серого ящика

- Grey Box Testing

Симбиоз использования подходов "Черный ящик" и "Белый ящик" увеличивает **покрытие возможных сценариев**

- **количественно,** потому что появляется большее количество тест-кейсов;
- **качественно,** потому что ПО тестируется принципиально разными подходами: с точки зрения пользователя ("Черный ящик") и с точки зрения кода ("Белый ящик").

Это подход, сочетающий элементы двух предыдущих подходов, это

- *с одной стороны,* тестирование, ориентированное на пользователя, а значит, мы используем паттерны поведения пользователя, т.е. **применяем методику "Черного ящика";**
- *с другой* — **информированное тестирование,** т.е. мы знаем, как устроена хотя бы часть тестируемого кода, и активно **используем** это знание.

Виды тестирования

1) По знанию внутренностей системы
2) По объекту тестирования
3) По субъекту тестирования
4) По времени проведения тестирования
5) По критерию “позитивности” сценариев
6) По степени автоматизированности тестирования
7) По степени подготовленности к тестированию

2. По объекту тестирования

- Функциональное тестирование
- Тестирование пользовательского интерфейса
- Тестирование локализации
- Тестирование скорости и надежности
- Тестирование безопасности
- Тестирование удобства использования
- Тестирование совместимости
- Тестирование инсталляции
- Тестирование документации

Функциональные виды тестирования

Функциональные тесты базируются на функциях и особенностях, а также взаимодействии с другими системами, и могут быть представлены на всех уровнях тестирования.

Функциональные виды тестирования рассматривают внешнее поведение системы.

1. Функциональное тестирование (Functional Testing)

Функциональное тестирование рассматривает заранее указанное поведение и основывается на анализе спецификаций функциональности компонента или системы в целом.

Функциональные тесты основываются на функциях, выполняемых системой, и могут проводиться на всех уровнях тестирования (*компонентном, интеграционном, системном, приемочном*). Как правило, эти функции описываются в требованиях, функциональных спецификациях или в виде вариантов использования системы (**use cases**).

Тестирование функциональности может проводится в двух аспектах:

- требования
- бизнес-процессы

1. Функциональное тестирование (Functional Testing)

Преимущества функционального тестирования:

- имитирует фактическое использование системы;

Недостатки функционального тестирования:

- возможность упущения логических ошибок в программном обеспечении;
- вероятность избыточного тестирования.

Достаточно распространенной является автоматизация функционального тестирования.

2. Тестирование безопасности (Security and Access Control Testing)

Тестирование безопасности - это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.

Принципы безопасности программного обеспечения

Общая стратегия безопасности основывается на трех основных принципах:

- конфиденциальность
- целостность
- доступность

Принципы безопасности ПО

- ***Конфиденциальность***

Конфиденциальность - это сокрытие определенных ресурсов или информации. Под конфиденциальностью можно понимать ограничение доступа к ресурсу некоторой категории пользователей, или другими словами, при каких условиях пользователь авторизован получить доступ к данному ресурсу.

- ***Целостность***

Существует два основных критерия при определении понятия целостности:

1. **Доверие**. Ожидается, что ресурс будет изменен только соответствующим способом определенной группой пользователей.
2. **Повреждение и восстановление**. В случае когда данные повреждаются или неправильно меняются авторизованным или не авторизованным пользователем, вы должны определить на сколько важной является процедура восстановления данных.

- ***Доступность***

Доступность представляет собой требования о том, что ресурсы должны быть доступны авторизованному пользователю, внутреннему объекту или устройству. Как правило, чем более критичен ресурс тем выше уровень доступности должен быть.

Виды уязвимостей

- **XSS (Cross-Site Scripting)** - это вид уязвимости программного обеспечения (Web приложений), при которой, на генерированной сервером странице, выполняются вредоносные скрипты, с целью атаки клиента.
- **XSRF / CSRF (Request Forgery)** - это вид уязвимости, позволяющий использоваться недостатки HTTP протокола, при этом злоумышленники работают по следующей схеме: ссылка на вредоносный сайт устанавливается на странице, пользующейся доверием у пользователя, при переходе по вредоносной ссылке выполняется скрипт, сохраняющий личные данные пользователя (пароли, платежные данные и т.д.), либо отправляющий СПАМ сообщения от лица пользователя, либо изменяет доступ к учетной записи пользователя, для получения полного контроля над ней.
- **Code injections (SQL, PHT, ASP и т.д.)** - это вид уязвимости, при котором становится возможно осуществить запуск исполняемого кода с целью получения доступа к системным ресурсам, несанкционированного доступа к данным либо выведения системы из строя.
- **Server-Side Includes (SSI) Injection** - это вид уязвимости, использующий вставку серверных команд в HTML код или запуск их напрямую с сервера.
- **Authorization Bypass** - это вид уязвимости, при котором возможно получить несанкционированный доступ к учетной записи или документам другого пользователя

3. Тестирование взаимодействия (Interoperability Testing)

Тестирование взаимодействия – это функциональное тестирование, проверяющее способность приложения взаимодействовать с одним и более компонентами или системами и включающее в себя тестирование совместимости (compatibility testing) и интеграционное тестирование (integration testing).

Программное обеспечение с хорошими характеристиками взаимодействия может быть легко интегрировано с другими системами, не требуя каких-либо серьезных модификаций. В этом случае, количество изменений и время, требуемое на их выполнение, могут быть использованы для измерения возможности взаимодействия.

Нефункциональные виды тестирования

- Нагрузочное тестирование
- Тестирование Установки или Installation Testing
- Тестирование удобства пользования или Usability Testing
- Тестирование на отказ и восстановление или Failover and Recovery Testing
- Конфигурационное тестирование или Configuration Testing

Нагрузочное тестирование

Нагрузочное тестирование или **тестирование производительности** - это автоматизированное тестирование, имитирующее работу определенного количества бизнес пользователей на каком либо общем (разделяемом ими) ресурсе.

Виды нагрузочного тестирования:

- **Тестирование производительности** (**Performance testing**)
- **Стрессовое тестирование** (**Stress Testing**)
- **Объемное тестирование** (**Volume Testing**)
- **Тестирование стабильности или надежности** (**Stability / Reliability Testing**)

Тестирование производительности

Задачей тестирования производительности является определение масштабируемости приложения под нагрузкой, при этом происходит:

- измерение времени выполнения выбранных операций при определенных интенсивностях выполнения этих операций
- определение количества пользователей, одновременно работающих с приложением
- определение границ приемлемой производительности при увеличении нагрузки (при увеличении интенсивности выполнения этих операций)
- исследование производительности на высоких, предельных, стрессовых нагрузках

Стрессовое тестирование

Стрессовое тестирование позволяет проверить насколько приложение и система в целом работоспособны в условиях стресса и также оценить способность системы к регенерации, т.е. к возвращению к нормальному состоянию после прекращения воздействия стресса.

Стрессом в данном контексте может быть повышение интенсивности выполнения операций до очень высоких значений или аварийное изменение конфигурации сервера.

Также одной из задач при стрессовом тестировании может быть оценка деградации производительности, таким образом цели стрессового тестирования могут пересекаться с целями тестирования производительности.

Объемное тестирование

Задачей объемного тестирования является получение оценки производительности при увеличении объемов данных в базе данных приложения, при этом происходит:

- измерение времени выполнения выбранных операций при определенных интенсивностях выполнения этих операций
- может производиться определение количества пользователей, одновременно работающих с приложением

Тестирование стабильности

Задачей тестирования стабильности (надежности) является проверка работоспособности приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки.

Времена выполнения операций могут играть в данном виде тестирования второстепенную роль.

При этом на первое место выходит отсутствие утечек памяти, перезапусков серверов под нагрузкой и другие аспекты влияющие именно на стабильность работы.

Тестирование Установки

Тестирование установки направленно на проверку успешной инсталляции и настройки, а также обновления или удаления программного обеспечения.

1. Проверка инструкции для инсталляции.
2. В распределенных системах требуется план установки с предусмотренным откатом. Его тоже нужно тестировать.

Тестирование установки можно назвать одной из важнейших задач по обеспечению качества программного обеспечения.

Особенности инсталляторов

Инсталлятор - это "обычная" программа, основные функции которой - Установка (Инсталляция), Обновление и Удаление (Деинсталляция) программного обеспечения.

Являясь обычной программой, инсталлятор обладает рядом особенностей, среди которых стоит отметить следующие:

- Глубокое взаимодействие с операционной системой и зависимость от неё (файловая система, реестр, сервисы и библиотеки)
- Совместимость как родных, так и сторонних библиотек, компонент или драйверов, с разными платформами
- Удобство использования: интуитивно понятный интерфейс, навигация, сообщения и подсказки
- Дизайн и стиль инсталляционного приложения
- Совместимость пользовательских настроек и документов в разных версиях приложения

Особенности инсталляторов

Список рисков, который покажет всю значимость корректной работы инсталляторов:

- риск потери пользовательских данных
- риск вывода операционной системы из строя
- риск неработоспособности приложения
- риск не корректной работы приложения

Тестирование удобства пользования

Тестирование удобства пользования - это метод тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий.

Уровни проведения

- Проверка удобства использования может проводиться как по отношению к готовому продукту, посредством тестирования черного ящика,
- так и к интерфейсам приложения (API), используемым при разработке - тестирование белого ящика.

Тестирование удобства пользования

Тестирование удобства пользования дает оценку уровня удобства использования приложения по следующим пунктам:

- **производительность, эффективность** (**efficiency**) - сколько времени и шагов понадобится пользователю для завершения основных задач приложения, например, размещение новости, регистрации, покупка и т. д.? (*меньше - лучше*)
- **правильность** (**accuracy**) - сколько ошибок сделал пользователь во время работы с приложением? (*меньше - лучше*)
- **активизация в памяти** (**recall**) – как много пользователь помнит о работе приложения после приостановки работы с ним на длительный период времени? (*повторное выполнение операций после перерыва должно проходить быстрее чем у нового пользователя*)
- **эмоциональная реакция** (**emotional response**) – как пользователь себя чувствует после завершения задачи - растерян, испытал стресс? Порекомендует ли пользователь систему своим друзьям? (*положительная реакция - лучше*)

Тестирование на отказ и восстановление

проверяет тестируемый продукт с точки зрения способности противостоять и успешно восстанавливаться после возможных сбоев, возникших в связи с ошибками программного обеспечения, отказами оборудования или проблемами связи (например, отказ сети).

Целью данного **вида тестирования** является проверка систем восстановления (или дублирующих основной функционал систем), которые, в случае возникновения сбоев, обеспечат сохранность и целостность данных тестируемого продукта.

Тестирование на отказ и восстановление очень важно для систем, работающих по принципу “24x7”.

Методика подобного **тестирования** заключается в симулировании различных условий сбоя и последующем изучении и оценке реакции защитных систем.

Тестирование на отказ и восстановление

Объектом тестирования в большинстве случаев являются весьма вероятные эксплуатационные проблемы, такие как:

- Отказ электричества на компьютере-сервере
- Отказ электричества на компьютере-клиенте
- Незавершенные циклы обработки данных (прерывание работы фильтров данных, прерывание синхронизации).
- Объявление или внесение в массивы данных невозможных или ошибочных элементов.
- Отказ носителей данных.

Тестирование на отказ и восстановление

Технически реализовать тесты можно следующими путями:

- Симулировать внезапный отказ электричества на компьютере (обесточить компьютер).
- Симулировать потерю связи с сетью (выключить сетевой кабель, обесточить сетевое устройство)
- Симулировать отказ носителей (обесточить внешний носитель данных)
- Симулировать ситуацию наличия в системе неверных данных (специальный тестовый набор или база данных).

Во всех вышеперечисленных случаях, по завершении процедур восстановления, должно быть достигнуто определенное требуемое состояние данных продукта:

- Потеря или порча данных в допустимых пределах.
- Отчет или система отчетов с указанием процессов или транзакций, которые не были завершены в результате сбоя.

Конфигурационное тестирование

Конфигурационное тестирование— специальный вид тестирования, направленный на проверку работы программного обеспечения при различных конфигурациях системы (заявленных платформах, поддерживаемых драйверах, при различных конфигурациях компьютеров и т.д.)

В зависимости от типа проекта конфигурационное тестирование может иметь разные цели:

- Проект по профилированию работы системы **Цель Тестирования**: определить оптимальную конфигурацию оборудования, обеспечивающую требуемые характеристики производительности и времени реакции тестируемой системы.
- Проект по миграции системы с одной платформы на другую **Цель Тестирования**: Проверить объект тестирования на совместимость с объявленным в спецификации оборудованием, операционными системами и программными продуктами третьих фирм.

Конфигурационное тестирование

Уровни проведения тестирования

Для клиент-серверных приложений конфигурационное тестирование можно условно разделить на два уровня (для некоторых типов приложений может быть актуален только один):

- Серверный
- Клиентский

На **первом (серверном)** уровне, тестируется взаимодействие выпускаемого программного обеспечения с окружением, в которое оно будет установлено:

- Аппаратные средства (тип и количество процессоров, объем памяти, характеристики сети / сетевых адаптеров и т.д.)
- Программные средства (ОС, драйвера и библиотеки, стороннее ПО, влияющее на работу приложения и т.д.)

Основной упор здесь делается на тестирование с целью определения оптимальной конфигурации оборудования, удовлетворяющего требуемым характеристикам качества.

Конфигурационное тестирование

На **следующем (клиентском)** уровне, программное обеспечение тестируется с позиции его конечного пользователя и конфигурации его рабочей станции. На этом этапе будут протестированы следующие характеристики: удобство использования, функциональность. Для этого необходимо будет провести ряд тестов с различными конфигурациями рабочих станций:

- Тип, версия и битность операционной системы (подобный вид тестирования называется **кросс-платформенное тестирование**)
- Тип и версия Web барузера, в случае если тестируется Web приложение (подобный вид тестирования называется **кросс-браузерное тестирование**)
- Тип и модель видео адаптера (при тестировании игр это очень важно)
- Работа приложения при различных разрешениях экрана
- Версии драйверов, библиотек и т.д. (для JAVA приложений версия JAVA машины очень важна, тоже можно сказать и для .NET приложений касательно версии .NET библиотеки)

Конфигурационное тестирование

Порядок проведения тестирования

Перед началом проведения конфигурационного тестирования рекомендуется:

- создавать матрицу покрытия (**матрица покрытия** - это таблица, в которую заносят все возможные конфигурации),
- проводить приоритезацию конфигураций (на практике, скорее всего, все желаемые конфигурации проверить не получится),
- шаг за шагом, в соответствии с расставленными приоритетами, проверяют каждую конфигурацию.

Тестирование пользовательского интерфейса

- Цель – проверить, насколько легко конечный пользователь системы может ее освоить, включая не только функциональную составляющую – саму систему, но и ее документацию; насколько эффективно пользователь может выполнять задачи, автоматизация которых осуществляется с использованием данной системы; наконец, насколько хорошо система застрахована (с точки зрения потенциальных сбоев) от ошибок пользователя.

- Для стандартных объектов UI
  - Text box
  - Text area
  - Numeric field
  - Drop down list
  - Combo box
  - Check box
  - Date field
- существуют стандартные проверки…

Тестирование локализации

- это процесс тестирования локализованной версии программного продукта. Проверка правильности перевода элементов интерфейса пользователя, проверка правильности перевода системных сообщений и ошибок, проверка перевода раздела "Помощь"/" Справка" и сопроводительной документации.
- С помощью тестирования локализации проверяются перевод, вспомогательные файлы, правильное обоснование и адаптация элементов интерфейса, а также правила написания текста.

Виды тестирования

1) По знанию внутренностей системы
2) По объекту тестирования
3) По субъекту тестирования
4) По времени проведения тестирования
5) По критерию “позитивности” сценариев
6) По степени автоматизированности тестирования
7) По степени подготовленности к тестированию

3. По субъекту тестирования

- Альфа-тестирование

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

- Бета-тестирование

Вид тестирования, который производится обычно потенциальными пользователями (проверка в “боевых условиях”).Иногда бета-тестирование выполняется для того, чтобы получить обратную связь о продукте от его будущих пользователей.

Виды тестирования

1) По знанию внутренностей системы
2) По объекту тестирования
3) По субъекту тестирования
4) По времени проведения тестирования
5) По критерию “позитивности” сценариев
6) По степени автоматизированности тестирования
7) По степени подготовленности к тестированию

4. По времени проведения тестирования

- Тест приемки (Smoke test)
- Тестирование новых функций
- Регрессионное тестирование
- Тест сдачи (Acceptance test)
- Эксплуатационное тестирование (Maintenance testing)

Smoke testing

применяется для поверхностной проверки всех модулей приложения на предмет работоспособности и наличия быстро находимых критических и блокирующих дефектов.

Подвидом дымового тестирования являются тестирование сборки, выполняемые на функциональном уровне командой тестирования, по результатам которого делается вывод о том, принимается или нет установленная версия программного обеспечения в тестирование, эксплуатацию или на поставку заказчику.

Для облегчения работы, экономии времени и людских ресурсов рекомендуется внедрить автоматизацию тестовых сценариев для дымового тестирования.

New feature testing – «Тестирование новой функции»

- Целью является проверка того, что новая функциональность работает корректно

Регрессионное тестирование

- Regression testing – повторное проведение тестов для проверки того, что изменения, внесенные в программу, не повлияли на функционал, который не изменялся.

- Не путать с retesting!

Full regression test

- Полный регрессионный тест включает в себя все ранее уже проверенное
- Может проводится на этапе, когда продукт уже заявлен как готовый к поставке
- Может включать в себя перепроверки ранее описанных багов

Acceptance testing
(приемо-сдаточное тестирование)

- Это набор тестов, по которому осуществляется приемка продукта (возможно заказчиком).
- **Решение о проведении приемочного тестирования** принимается, когда:
  - продукт достиг необходимого уровня качества;
  - заказчик ознакомлен с **Планом Приемочных Работ** (**Product Acceptance Plan**) или иным документом, где описан набор действий, связанных с проведением приемочного тестирования, дата проведения, ответственные и т.д.

Maintenance testing

- Тестирование системы во время ее эксплуатации

Sanity testing

- **Санитарное тестирование** - это узконаправленное тестирование достаточное для доказательства того, что конкретная функция работает согласно заявленным в спецификации требованиям.
- Является подмножеством регрессионного тестирования.
- Используется для определения работоспособности определенной части приложения после изменений произведенных в ней или окружающей среде.
- Обычно выполняется вручную.
- В отличии от дымового, **санитарное тестирование направлено вглубь** проверяемой функции, в то время как **дымовое направлено вширь**, для покрытия тестами как можно большего функционала в кратчайшие сроки.

Виды тестирования

1) По знанию внутренностей системы
2) По объекту тестирования
3) По субъекту тестирования
4) По времени проведения тестирования
5) По критерию “позитивности” сценариев
6) По степени автоматизированности тестирования
7) По степени подготовленности к тестированию

5. По критерию “позитивности” сценариев

- Позитивное тестирование
- Негативное тестирование

Позитивное тестирование

- Класс тестов, которые проверяют, что программа делает то, что должна делать (результатом ожидается УСПЕХ)

Негативное тестирование

- Класс тестов, которые проверяют, что программа НЕ делает то, что НЕ должна делать (ожидается НЕУСПЕХ в результате)

Виды тестирования

1) По знанию внутренностей системы
2) По объекту тестирования
3) По субъекту тестирования
4) По времени проведения тестирования
5) По критерию “позитивности” сценариев
6) По степени автоматизированности тестирования
7) По степени подготовленности к тестированию

6. По автоматизированности тестирования

- Ручное
- Автоматизированное
- Смешанное\полуавтоматическое

Ручное тестирование

- Выполняется без привлечения средств автоматизации
- Выполняется, обычно, по подготовленным тест кейсам

Автоматизированное тестирование

- Выполняется с использованием специализированных программных продуктов
- Требуется высокая квалификация тестировщиков и навыки программирования

Виды тестирования

1) По знанию внутренностей системы
2) По объекту тестирования
3) По субъекту тестирования
4) По времени проведения тестирования
5) По критерию “позитивности” сценариев
6) По степени автоматизированности тестирования
7) По степени подготовленности к тестированию

7. По подготовленности к тестированию

- Тестирование по тест кейсам (documented testing)
- Интуитивное тестирование (ad hoc testing)
- Исследовательское тестирование (exploratory testing)
- Обезьянье тестирование(monkey testing)

Тестирование по тест кейсам (documented testing)

- Тестирование по уже разработанной тестовой документации
- Выполняемые тесты определены заранее

Интуитивное тестирование (ad hoc testing)

- Тестирование может происходить без сценария, когда тестировщик бессистемно перебирает различные варианты работы системы

Исследовательское тестирование (exploratory testing)

- Исследовательское Тестирование — одновременно является и техникой и видом тестирования. Такое тестирование подразумевает под собой одновременно изучение проекта, функционала, проектирование тест кейсов в уме и тут же их исполнение не записывая и не создавая тестовую документацию.

Обезьянье тестирование(monkey testing)

- Monkey testing – автоматизированный (не обязательно) тест, который выполняется без специфического, четко определенного тестового сценария. Название является метафорическим, имеется в виду, что операции ввода данных является случайные и бессмысленные, как будто их выполняет обезьяна. Например, обезьяний тест может вводить произвольные строки в поля ввода, чтобы проверить корректность работы системы в случае введения любых возможных данных. Данные введения могут быть заданы заранее, случайными генерируемыми, генерируемыми с учетом статистики пользователя