Лекция 3. Классификация видов тестирования ПО

продолжение

Уровни тестирования

Компонентное (модульное) тестирование

– это тестирование программы на уровне отдельно взятых модулей, функций или классов.

- *Цель модульного тестирования* – выявление локализованных в модуле ошибок в реализации алгоритмов, а также определение степени готовности системы к переходу на следующий уровень разработки и тестирования.
- Модульное тестирование проводится по принципу "белого ящика", то есть основывается на знании внутренней структуры программы, и часто включает те или иные методы анализа покрытия кода.

Модульное тестирование проверяет функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы по отдельности (модули программ, объекты, классы, функции и т.д.).

Обычно компонентное (модульное) тестирование проводится вызывая код, который необходимо проверить и при поддержке сред разработки, таких как фреймворки (frameworks - каркасы) для модульного тестирования или инструменты для отладки. Все найденные дефекты, как правило исправляются в коде без формального их описания в системе менеджмента багов.

Один из наиболее эффективных подходов к компонентному (модульному) тестированию - это подготовка автоматизированных тестов до начала основного кодирования (разработки) программного обеспечения. Это называется разработка от тестирования или подход тестирования вначале.

При этом подходе создаются и интегрируются небольшие куски кода, напротив которых запускаются тесты, написанные до начала кодирования. Разработка ведется до тех пор, пока все тесты не будут успешно пройдены.

Интеграционное тестирование

– это тестирование части системы, состоящей из двух и более модулей.

- *Цель интеграционного тестирования* – поиск дефектов, связанных с ошибками в реализации и интерпретации интерфейсного взаимодействия между модулями.
- Интеграционное тестирование оперирует интерфейсами модулей и подсистем и требует создания тестового окружения, включая заглушки (Stub) на месте отсутствующих модулей.

- **Интеграционное тестирование** предназначено для проверки связи между компонентами, а также взаимодействия с различными частями системы (операционной системой, оборудованием либо связи между различными системами).

Уровни интеграционного тестирования:

- **Компонентный интеграционный уровень** - проверяется взаимодействие между компонентами системы после проведения компонентного тестирования.
- **Системный интеграционный** - проверяется взаимодействие между разными системами после проведения системного тестирования.

тестированию:

- **Снизу вверх: в**се низкоуровневые модули собираются воедино и затем тестируются. После чего собирается следующий уровень модулей для проведения интеграционного тестирования. Данный подход считается полезным, если все или практически все модули разрабатываемого уровня готовы.
- **Сверху вниз:** сначала тестируются все высокоуровневые модули, затем постепенно, один за другим, добавляются низкоуровневые. Все модули более низкого уровня симулируются заглушками с аналогичной функциональностью, затем, по мере готовности, они заменяются реальными активными компонентами.
- **Большой взрыв:** все (или практически все) разработанные модули собираются вместе в виде законченной системы или ее основной части, а затем проводится интеграционное тестирование.

Системное тестирование

– рассматривает тестируемую систему в целом и оперирует на уровне пользовательских интерфейсов.

- *Цель системного тестирования* – выявление дефектов, связанных с работой системы в целом, таких как неверное использование ресурсов системы, непредусмотренные комбинации данных пользовательского уровня, несовместимость с окружением, непредусмотренные сценарии использования, отсутствующая или неверная функциональность, неудобство в применении и тому подобное.
- Системное тестирование производится над проектом в целом с помощью метода «черного ящика».

Основной задачей **системного тестирования** является проверка как функциональных, так и не функциональных требований, дефекты в системе в целом.

При этом выявляется неверное использование ресурсов системы, непредусмотренные комбинации данных пользовательского уровня, несовместимость с окружением, непредусмотренные сценарии использования, отсутствующая или неверная функциональность, неудобство использования и т.д.

Для минимизации рисков, связанных с особенностями поведения системы в той или иной среде, во время тестирования рекомендуется использовать окружение, максимально приближенное к тому, на которое будет установлен продукт после выдачи.

Подходы к системному тестированию:

- **на базе требований** - для каждого требования пишутся тестовые случаи (test cases), проверяющие выполнение данного требования.
- **на базе случаев использования** - на основе представления о способах использования продукта создаются случаи использования системы (Use Cases). По конкретному случаю использования можно определить один или более сценариев. На проверку каждого сценария пишутся тест-кейсы, которые должны быть протестированы.

Виды тестирования

Все виды тестирования программного обеспечения, в зависимости от преследуемых целей, можно условно разделить на следующие группы:

- Функциональные.
- Нефункциональные.
- Связанные с изменениями.

Функциональные виды тестирования

Функциональные тесты базируются на функциях и особенностях, а также на взаимодействии с другими системами и могут быть представлены на всех уровнях тестирования: модульном, интеграционном, системном. Функциональные виды тестирования рассматривают внешнее поведение системы.

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

- Функциональные тесты
- Тестирование безопасности
- Тестирование взаимодействия

Тестирование функциональности может проводиться в двух **аспектах**:

- **Требования**. Использует спецификацию функциональных требований к системе, как основу для дизайна тестовых случаев. Необходимо сделать список того, что будет тестироваться, приоритезировать требования на основе рисков, а потом приоритезировать тестовые сценарии . Это позволит сфокусироваться и не упустить при тестировании наиболее важный функционал.
- **Бизнес-процессы**. Тестирование в аспекте «бизнес-процессы» использует знание бизнес-процессов, которые описывают сценарии ежедневного использования системы. В этом аспекте тестовые сценарии, как правило, основываются на случаях использования системы.

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.

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:

- Конфиденциальность.
- Целостность.
- Доступность.

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

Критерии определения понятия целостности:

- **Доверие.** Ожидается, что ресурс будет изменен только соответствующим способом определенной группой пользователей.
- **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.

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

Программное обеспечение с хорошими характеристиками взаимодействия может быть легко интегрировано с другими системами, не требуя каких-либо серьезных модификаций.

Нефункциональные виды тестирования

Нефункциональное тестирование описывает тесты, необходимые для определения характеристик программного обеспечения, которые могут быть измерены различными величинами. В целом, это тестирование того, как система работает.

Нефункциональное тестирование – проверка корректности работы нефункциональных требований. Оценивается, КАК программный продукт работает.

Виды нефункционального тестирования:

- Тестирование производительности – работа ПО под определённой нагрузкой.
- Тестирование пользовательского интерфейса – удобство пользователя при взаимодействии с разными параметрами интерфейса (кнопки, цвета, выравнивание и т. д.).
- Тестирование UX – правильность логики использования программного продукта.
- Тестирование защищенности – определение безопасности ПО: защищено ли оно от атак, несанкционированного доступа к данным и т.д.

- Инсталляционное тестирование – оценка вероятности возникновения проблем при установке, удалении, а также обновлении ПО.
- Тестирование совместимости – тестирование работы программного продукта в определённом окружении.
- Тестирование надежности – работа программы при длительной средней ожидаемой нагрузке.
- Тестирование локализации – оценка правильности версии программного продукта (языковой и культурный аспекты).

производительности

- **Тестирование производительности** - определение масштабируемости приложения под нагрузкой: определение количества пользователей, одновременно работающих с приложением; определение границ приемлемой производительности при увеличении нагрузки; исследование производительности на высоких, предельных, стрессовых нагрузках.
- **Стрессовое тестирование** - позволяет проверить, насколько приложение и система в целом работоспособны в условиях стресса, а также оценить способность системы к регенерации. Стрессом может быть повышение интенсивности выполнения операций до очень высоких значений или аварийное изменение конфигурации сервера.

- **Объемное тестирование** - получение оценки производительности при увеличении объемов данных в базе данных приложения: измерение времени выполнения выбранных операций при определенных интенсивностях выполнения этих операций; определение количества пользователей, одновременно работающих с приложением.
- **Тестирование стабильности или надежности** - проверка работоспособности приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки. Время выполнения операций может 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

Направленно на проверку успешной инсталляции и настройки, а также на обновление или удаление программного обеспечения.

- Установка ПО при помощи инсталляторов.
- Самостоятельная установка ПО, используя документацию в виде инструкций или "readme" файлов.

Тестирование удобства пользования

Для того, чтобы приложение было популярным, ему мало быть функциональным – оно должно быть еще и удобным.

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 удобства использования приложения:

- **Производительность, эффективность** - сколько времени и шагов нужно пользователю для завершения основных задач приложения, например, регистрации и т.д. (меньше - лучше).
- **Правильность** - сколько ошибок сделал пользователь во время работы с приложением (меньше - лучше).
- **Активизация в памяти** – 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** – как пользователь себя чувствует после завершения задачи? 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).

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

Очень важно для систем, работающих по принципу “24x7”.

Методика тестирования заключается в симулировании различных условий сбоя и последующем изучении и оценке реакции защитных систем. В процессе подобных проверок выясняется, была ли достигнута требуемая степень восстановления системы после возникновения сбоя.

Объектом тестирования обычно являются весьма вероятные эксплуатационные проблемы, такие как:

- Отказ электричества на компьютере-сервере или 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.

При достижении соответствующих условий сбоя и по результатам работы систем восстановления, можно оценить продукт с точки зрения тестирования на отказ. 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

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

Цели тестирования:

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

Виды тестирования, связанные с изменениями ПО

После проведения необходимых изменений в ПО (исправление багов/дефектов), оно должно быть перетестировано для подтверждения того факта, что проблема была действительно решена.

Виды тестирования:

- Дымовое тестирование
- Регрессионное тестирование
- Тестирование сборки
- Санитарное тестирование или проверка согласованности/исправности

Дымовое тестирование

Дымовое тестирование рассматривается как короткий цикл тестов, выполняемый для подтверждения того, что, после сборки кода (нового или исправленного), устанавливаемое приложение стартует и выполняет основные функции.

Тестируются наиболее важные модули приложения на предмет возможности выполнения требуемых задач и наличия быстронаходимых критических и блокирующих дефектов.

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

Регрессионное тестирование

- проверка изменений, сделанных в приложении или окружающей среде (починка дефекта, слияние кода, миграция на другую ОС, БД, веб-сервер или сервер приложения), для подтверждения того факта, что существующая ранее функциональность работает как и прежде. Регрессионными могут быть как функциональные, так и нефункциональные тесты.

Как правило, для регрессионного тестирования используются тест-кейсы, написанные на ранних стадиях разработки и тестирования.

Рекомендуется делать автоматизацию регрессионных тестов для ускорения последующего процесса тестирования и обнаружения дефектов на ранних стадиях разработки ПО.

Тестирование сборки

Тестирование, направленное на определение соответствия выпущенной версии критериям качества для начала тестирования.

По своим целям является аналогом дымового тестирования, направленного на приемку новой версии в дальнейшее тестирование или эксплуатацию. Вглубь оно может проникать дальше, в зависимости от требований к качеству выпущенной версии.

Санитарное тестирование или проверка согласованности/исправности

- 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.