ЛАБОРАТОРНАЯ РАБОТА №2

Тема лабораторной работы: работа с классификацией видов тестирования.

Общие сведения по работе

Классификация видов тестирования отражает многообразие походов организации контроля качества программных систем. Понимание классификации позволяет определить возможные варианты процедур тестирования, оценить трудоемкость, спланировать тестирование, выбрать подходящие инструменты. Это знание является необходимым при построении жизненного цикла разработки программных систем. Навыки использования видов тестирования для контроля функционирования программных систем позволяют тестировщику быстрее локализовать дефекты, создавать более полные наборы тестов.

Методические рекомендации и материалы

Деление на различные виды тестирования позволяет применять только подходы соразмерно требованиям к проверке поставленных требований, т. е. отдельные виды тестирования могут получить повышенный приоритет выполнения, в то время как все прочие виды тестирования будут не так важны. Выбор видов тестирования зависит от поставленных требований (зачастую в ТЗ уже сформулированы требования приемочного тестирования, которые нужно будет проверить при готовности системы), этапа разработки, условий использования системы. Используемые виды тестирования можно использовать в чистом виде (например, тестирование пользовательского интерфейса), так и охарактеризовать с позиций типов, методов и других видов тестирования (ручное функциональное тестирование, автоматизированное функциональное тестирование, регрессионное ручное функциональное тестирование, и т. д.)

Представление классификации основано на делении следующим ключевым группам:

1. Уровни тестирования предполагают оценивать программную систему с точки зрения объема программных модулей и учета соответствующих аспектов (например, внимание будет сосредоточено только на проверке кода, реализующего бизнес-логику, или важным будет являться момент интеграции подсистем или модулей). В зависимости от степени детальности компонентов подразделяются на:

– Компонентное (модульное) – тестирование программы на уровне отдельно взятых модулей, функций или классов. Цель компонентного тестирования – выявление локализованных в модуле ошибок в реализации алгоритмов, а также определение степени готовности системы к переходу на следующий уровень разработки и тестирования.

– Интеграционное тестирование предназначено для проверки связи между компонентами, а также взаимодействия с различными частями системы (операционной системой, оборудованием либо связи между различными системами). Уровнями интеграционного тестирования являются: компонентный интеграционный (проверяется взаимодействие между компонентами системы после проведения компонентного тестирования), системный интеграционный уровень (проверяется взаимодействие между разными системами после проведения системного тестирования).

– Системное тестирование проверяет как функциональные, так и нефункциональные требования к системе в целом. При этом выявляются дефекты, такие как неверное использование ресурсов системы, не предусмотренные комбинации данных пользовательского уровня, несовместимость с окружением, непредусмотренные сценарии

использования, отсутствующая или неверная функциональность, неудобство использования и т. д.

При этом деление на уровни тестирования предполагает участие различных ролей участников команды разработки. Предполагается, что тесты уровня компонентного тестирования разрабатывает сам программист, контролируя самостоятельно качество своей работы или работы своих коллег. Интеграция модулей или общесистемное тестирование могут быть проведены уже другими участниками команды, знающими требования к программным интерфейсам модулей и подсистемам и имеющими возможность для манипуляции с ними.

2. Типы тестирования сосредоточены на делении видов тестирования, проверяющих наличие требуемой функциональности (функциональный тип тестирования, что система делает) и проверяющих режимы работы системы при выполнении пользователями своих задач (нефункциональный тип тестирования, как система это делает).

Функциональный тип тестирования включает следующие подвиды:

– функциональное тестирование;

– тестирование возможности взаимодействия (включает тестирование совместимости и интеграционное тестирование);

– тестирование безопасности.

Нефункциональный тип тестирования описывает такие подвиды, как:

– нагрузочное тестирование;

– стресс-тестирование;

– тестирование удобства пользования;

– тестирование надежности и т. д.

Примеры проведения нефункциональных тестов:

– общее исследование поведения системы при экстремальных нагрузках;

– экспертиза обработки ошибок и исключений;

– тестирование пропускной способности системы;

– изучение некоторых частей системы или ее компонентов в условиях непропорциональной нагрузки.

3. Методы тестирования описывают применение видов тестирования в зависимости от объекта тестирования. Методы подразделяются на статические (тестирующие такие объекты, как программный код, документацию, ресурсы проекта, все то, что может быть проверено без непосредственного выполнения программного кода, т. е. запуска программной системы) и динамические (тестирующие функциональность программной системы и режимы ее эксплуатации).

Подвидом статического метода тестирования является рецензирование – вид тестирования, который может проводиться перед динамическим тестированием.

То, как он может быть проведен:

– вручную,

– с применением анализаторов кода.

Рецензирование может проводится для любого продукта, связанного с разработкой программного обеспечения, включая спецификации требований и дизайна, код, планы тестирования, руководства пользователя и т.п. Во время рецензирования могут быть найдены упущения, например в требованиях, которые маловероятно найти во время динамического тестирования. Более широким видом статического тестирования является статический анализ, проводимый при помощи специальных инструментов. При этом анализируется:

– код программы (например, потоки управления и поток данных),

– сгенерированный код, например HTML, XML.

На рисунке 1 приведен пример скриншота отчета статического анализатора кода SonarQube с результатами анализа, охарактеризованных по степени серьезности, с рекомендацией к устранению. На рисунке 2 проводится отчет о валидации html разметки сайта ulstu.ru.

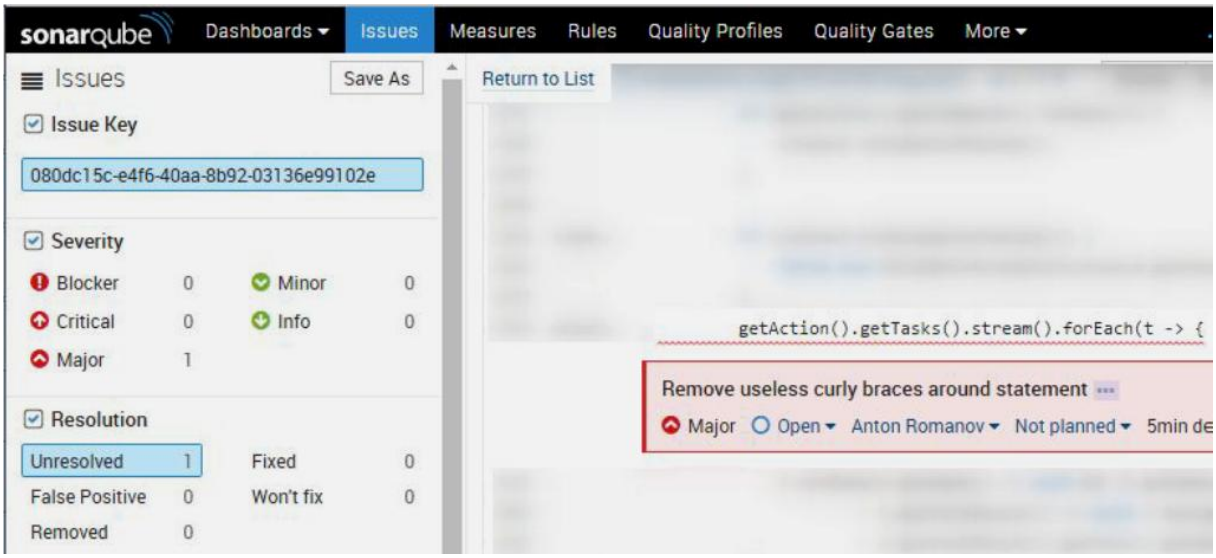

Рис. 1. Отчет в системе SonarQube о статическом анализе программного кода проекта

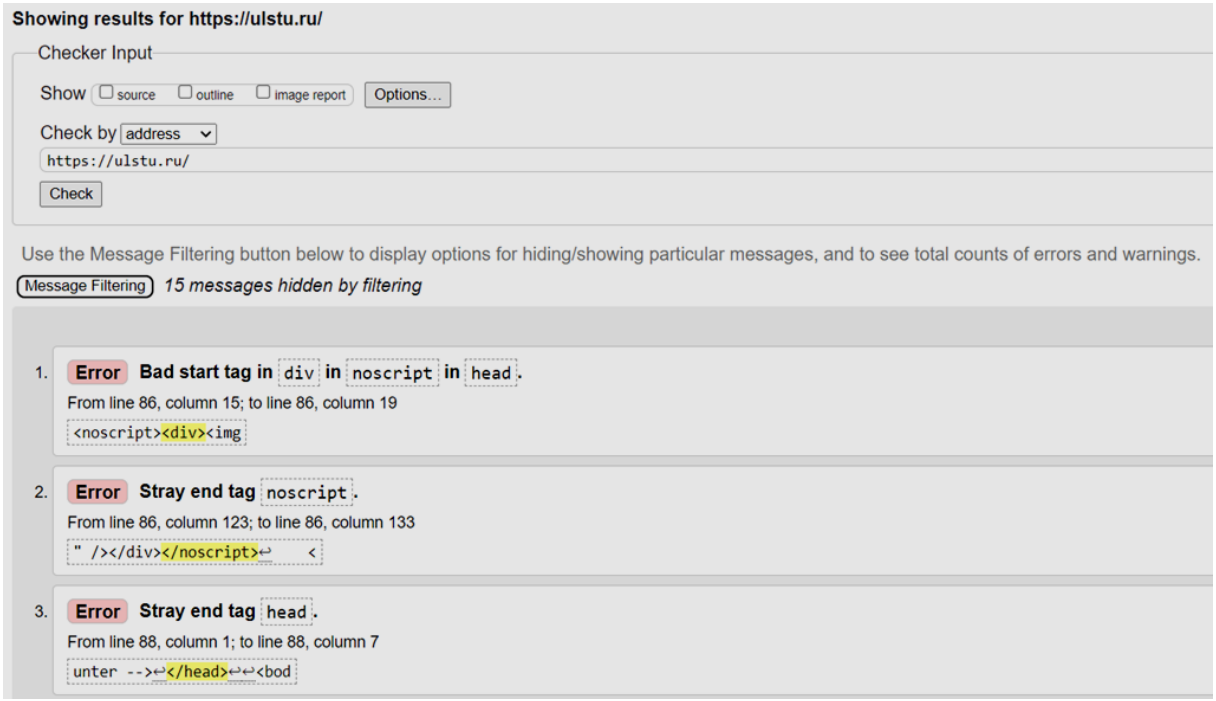

Рис. 2. Отчет валидации html разметки сайта

Динамические методы тестирования позволяют проверять программную систему при ее непосредственном выполнении. Динамические методы делятся на:

– Методы белого ящика. Тестирование на соответствие программного продукта требованиям со знанием внутренней структуры реализации системы (есть в наличии исходный код и технические спецификации). Это вид тестирования позволяет проводить локализацию ошибок, анализ надежности и устойчивости и другие типы проверок, существенно повышая качество системы.

– Методы черного ящика. Тестирование на соответствие программного продукта требованиям без знания внутренней структуры реализации системы.

– Методы серого ящика. Комбинирование методов белого и черного ящика. При тестировании серого ящика разработчик теста имеет доступ к исходному коду, но при непосредственном выполнении тестов доступ к коду, как правило, не требуется.

4. Виды тестирования, подразделяются на группы:

- по объекту (на что обращен процесс тестирования).
    - Функциональное. Проверяется соответствие требованиям. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывали на различных значениях.
    - Навигация. Действие кнопки «назад» браузера для сайта, ошибки: 404, 500 и т. д.
    - Инсталляция. Формальное тестирование программы установки (графический интерфейс пользователя, общее удобство пользования, соответствие стандартам), функциональное

тестирование программы установки, тестирование механизма лицензирования и способности противостоять взлому, проверка стабильности работы приложения после установки.

- Нагрузочное. Тесты производительности (количество обрабатываемых запросов в единицу времени), нагрузочные тесты (количество бизнес-пользователей приложения, генерирующих запросы в единицу времени), стресс-тесты (функционирование в условиях экстремальных уровней нагрузки и ограничения ресурсов, а также контроль за скоростью восстановления после стресса), объемное тестирование (увеличение объема хранимых, обрабатываемых данных).
- Ресурсные тесты. Тестирование устойчивости в течение длительного времени.
- Юзабилити. Тестирование удобства пользования приложением определяет, соответствует ли приложение потребностям целевой аудитории и отвечает ли оно требованиям пользователя (сколько шагов и времени нужно пользователям для выполнения их задач в приложении, частота возникновения проблем взаимодействия с пользовательским интерфейсом).
- UI/GUI. Тестирование на соответствие стандартам графических интерфейсов, тестирование с различными разрешениями экрана, тестирование в ограниченных условиях, например, в условиях нехватки памяти, тестирование локализованных версий: точность перевода, проверка длины названий элементов интерфейса, тестирование графического интерфейса пользователя на целевых устройствах.

- Локализация. Проверка правильности перевода согласно тематике данного сайта или программы. Проверка перевода раздела «Помощь» и сопроводительной документации.
- Безопасность. Тестирование безопасности представляет собой ряд услуг, от разработки политики безопасности до тестирования безопасности на уровне приложения, операционной системы и сетевой безопасности. Учитываются три главных критерия: конфиденциальность, целостность, доступность.
- Совместимость. Проверяется функционирование на различных операционных системах, платформах разработки, в различных браузерах.
- Конфигурация. Тестируется функционирование программных систем на различных аппаратных платформах и при различных конфигурациях аппаратно-зависимого программного обеспечения (драйвера, взаимодействие с периферийными устройствами).
- Документация. Тестируется на этапе разработки требований к программному продукту после создания функциональных спецификаций. Помогает избежать логических дефектов и ненужных изменений в продукте до начала его фактической разработки. Позволяет улучшить пользовательскую документацию.
- Прототип. Основная цель тестирования прототипа – выявить потенциальные проблемы в приложении, проверить, насколько приложение соответствует потребностям и ожиданиям пользователя, и обнаружить расхождения с требованиями к графическому интерфейсу пользователя. Проверяется:

структура приложения, формы пользовательского интерфейса, прототип бизнес-логики, логические связи между модулями, навигация, графический интерфейс пользователя;

- субъекту, степени автоматизации, признаку позитивности, времени проведения. Данная группа видов тестирования характеризует как выбор объекта тестирования и роль участника команды разработки, так и особенности условий реализации программной системы и жизненного цикла разработки:
    - Субъект тестирования – лицо или группа специалистов, проводящая тестирование (штатные работники, потенциальные заказчики на стороне разработчика, потенциальные пользователи готовой версии продукта с ограничениями).
    - По степени автоматизации: ручное тестирование, автоматизированное тестирование, полуавтоматизированное тестирование.
    - По признаку позитивности. Позитивное тестирование – это тестирование на данных или сценариях, которые соответствуют нормальному (штатному, ожидаемому) поведению системы. Основной целью позитивного тестирования является проверка того, что при помощи системы можно делать то, для чего она создавалась. Негативное тестирование – это тестирование на данных или сценариях, которые соответствуют нештатному поведению тестируемой системы – различные сообщения об ошибках, исключительные ситуации, запредельные состояния. Основной целью негативного тестирования является проверка устойчивости системы к воздействиям различного рода, валидация неверного набора данных, проверка обработки исключительных ситуаций

(как в реализации самих программных алгоритмов, так и в логике бизнес-правил).

- По времени проведения. Альфа-тестирование проводится на ранней стадии разработки продукта, но в некоторых случаях может применяться для законченного продукта в качестве внутреннего приемочного тестирования. Иногда альфа-тестирование выполняется под отладчиком или с использованием окружения, которое помогает быстро выявлять найденные ошибки. Тестирование новой функциональности проверяет корректность реализации новых задач. Тестирование при приемке (Smoke-testing, дымное тестирование) – поверхностная проверка всех модулей приложения на предмет работоспособности и наличия быстро находимых критических и блокирующих дефектов. Тестирование сборки направлено на определение соответствия выпущенной версии критериям качества для начала тестирования. Приемочное тестирование – формальный процесс тестирования, который проверяет соответствие системы требованиям и проводится с целью определения, удовлетворяет ли система приемочным критериям; вынесения решения заказчиком или другим уполномоченным лицом о приемке приложения. Регрессионное тестирование – это вид тестирования, направленный на проверку изменений, сделанных в приложении или окружающей среде (починка дефекта, слияние кода, миграция на другую операционную систему, базу данных, 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.

Задания к лабораторной работе

1. Выбрать ранее разработанный программный проект.
2. Провести анализ и составить отчет со следующей структурой:
    a. Цель работы.
    b. Описание программного проекта.
    c. По каждому из элементов классификации, описанному в разделе методических рекомендаций, привести список видов, типов, методов, уровней тестирования, применимых для выбранного проекта.
    d. Привести примеры дефектов, характерных для каждого вида тестирования.
    e. Выводы по работе.
    f. Список использованных источников.
3. Оформить и защитить отчет.

Контрольные вопросы

1. Определяется ли качество ПО качеством программного кода?
2. Какие существуют виды тестирования?
3. Какие существуют типы тестирования?
4. Какие существуют методы тестирования?
5. Какие существуют уровни тестирования?