СПбГЭТУ «ЛЭТИ»
ПЕРВЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ

Е. Л. Турнецкая

Организация среды тестирования

Конспект лекций

СПбГЭТУ «ЛЭТИ», 2023 г.

9. ОРГАНИЗАЦИЯ ПРОЦЕССА ТЕСТИРОВАНИЯ

9.3. ОРГАНИЗАЦИЯ СРЕДЫ ТЕСТИРОВАНИЯ

В системах управления тестированием фиксируют тест-кейсы и документы по тестированию, в баг-трекинговые системы заносят отчеты о найденных инцидентах, сопровождая их скриншотами и комментариями. Также тестировщику потребуются программы для документации в текстовом или табличном виде, программы для записи видео и создания скриншотов. Специалист по тестированию должен знать множество специализированных инструментов для проведения тестирования и выбирать конкретный для решения поставленной задачи.

В глоссарии ГОСТа ГОСТ Р 56920-2016/ISO/IEC/IEEE 29119-1:2013 «Системная и программная инженерия. Тестирование программного обеспечения. Часть 1. Понятия и определения», который создан на основе международных стандартов, дано следующее определение:

Процесс Выполнения Тестирования (Test Completion Process): Процесс менеджмента тестирования, необходимый для обеспечения доступности полезных активов тестирования для дальнейшего использования, обеспечения удовлетворительного состояния тестовых сред, гарантии документирования и передачи соответствующим заинтересованным сторонам результатов тестирования.

Команда тестировщиков отвечает за качество реализованного программного продукта. Для тестирования которого потребуются навыки работы в различных программных средах, начиная от простой фиксации скриншота экрана монитора и заканчивая, например, автоматизированным тестированием производительности и/или тестированием безопасности.

Компоненты среды реализации и тестирования:

1. Среда для создания программного кода.
2. Среда хранения версий программного кода и документации к проекту.
3. Среда создания тестовых наборов.
4. Тестовая среда для проведения проверок.
5. Система хранения тест-кейсов.
6. Баг-трекинговая система.
7. Система управления проектом.
8. Программные инструменты для логирования.
9. Программные инструменты для сбора тестовых доказательств и многие другие.

Тестовая среда

В процессе тестирование разворачивают тестовую среду. Она включает аппаратное обеспечение, измерительную аппаратуру, имитаторы, программный инструментарий и прочие инструменты, необходимые для проведения теста.

Тестовая среда (Test Env) – различные средства, аппаратное и программное обеспечение, встроенное микропрограммное обеспечение, процедуры и документация, предназначенные или используемые для выполнения тестирования программного обеспечения.

Тестирование в условиях, приближенным к реальным, проводят на испытательном стенде. На нем проведены настройки тестовой среды для проведения конкретных видов тестирования на подготовленных данных.

Испытательный стенд (Test Bed) – включает оборудование, операционную систему, конфигурацию сети, тестовые данные, которые полностью настроены для проведения тестирования ПО.

Программно-аппаратное окружение процесса тестирования

Рассмотрим типы программного окружения процесса тестирования:

– В среде разработки программисты пишут код, проводят отладку, исправляют ошибки, выполняют Unit-тестирование. За эту среду отвечают также разработчики.

– В среде тестирования проверяют новые сборки программного продукта: функционал, проводят регрессионные проверки. На этом этапе начинается динамическое тестирование, т.е. с запуском приложения на исполнение, тестирование.

– Интеграционная среда иногда реализована в рамках среды тестирования, а иногда в рамках превью среды. В этой среде собрана необходимая для end-to-end тестирования схема взаимодействующих друг с другом модулей, систем, продуктов.

– Preview среда, максимально приближенная к реальной (боевой): те же данные, то же аппаратно-программное окружение, та же производительность. Здесь тестировщики проводят заключительное end-to-end тестирование функционала.

– В среде production работают пользователи и специалисты поддержки.

Следует отметить, что в курсе рассмотрены бесплатные программные инструменты, которые внесены в реестр отечественного программного обеспечения или имеют открытый программный код (https://catalog.arppsoft.ru/section/2117).

Системы управления проектом

Управление тестированием относят к задачам управления проектом. Поэтому для организации процесса тестирования используют системы управления проектом.

Система управления проектами – это набор организационных и технологических методов и инструментов, которые поддерживают управление проектами в компании и помогают повысить эффективность их реализации.

Программные инструменты тестирования клиентской и серверной частей

Как правило, архитектура современного приложения построена с использованием шаблона MVC (Model-View-Controller) (рис.1). Поэтому можно базу данных можно протестировать параллельно интерфейсу приложения. Также многие веб-приложения построены на архитектурном паттерне «клиент-сервер», который также позволяет отдельно проводить тестовые мероприятия для клиентской и серверной стороны.

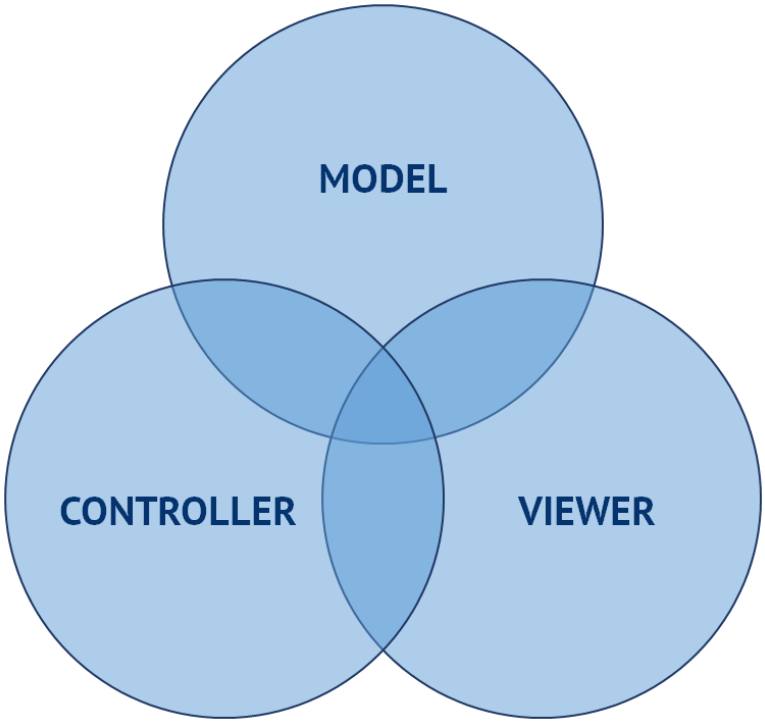

Рисунок 1 – Шаблон MVC

При тестировании клиентской части необходимо знать специальные инструменты, которые позволяю проверить верстку макета, соответствие дизайн-макету, функциональность каждого элемента, реализовать автоматизацию тестовых проверок. При тестировании фронтенда проводят ручное и автоматизированное тестирование.

Инструменты тестирования:

1. Selenium. Автоматизации действий веб-браузера (https://www.selenium.dev)
2. Браузерное расширение PerfectPixel. Проверка соответствия макету разработанной web-страницы (https://chrome.google.com/webstore/detail/perfectpixel-by-welldonec/dkaagdgjmgdmbnecmcefdhjekcoceebi)
3. Тестирование верстки с помощью валидатора W3C (https://validator.w3.org/)

Для проверки доступа к серверной части и ее функционирования применяют программные инструменты Postman, Katalon, JMeter. Как правило, на серверной части и при тестировании API проводят автоматизированное тесты (рис.2).

Рисунок 2 – Программные инструменты для тестирования серверной части

Непрерывная интеграция

Актуальным подходом к разработке программного обеспечения является непрерывная интеграция. Она позволяет выполнять отправку, сборку и тестирование нового кода по мере его написания (рис.3).

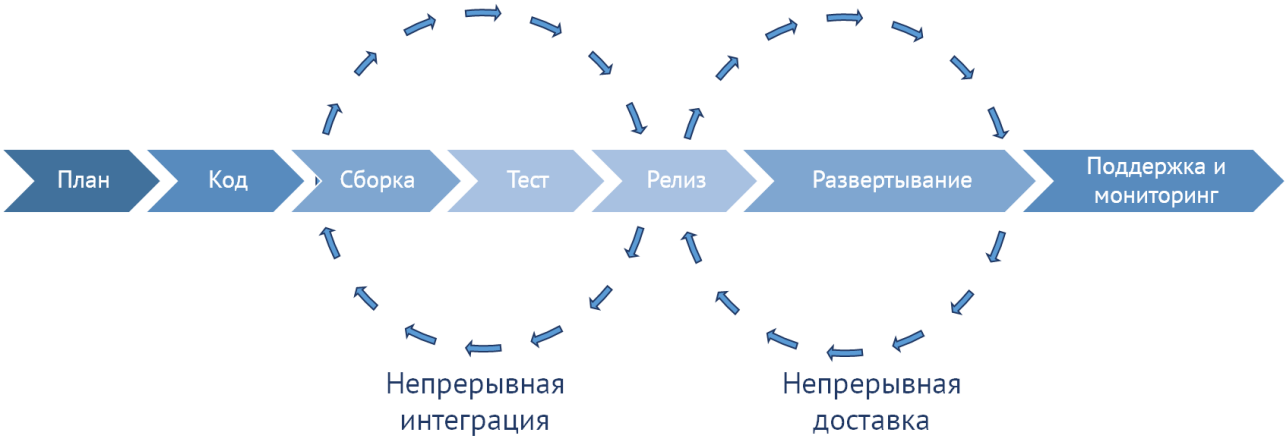

Рисунок 3 – Непрерывная интеграция

Идея интеграции состоит в следующем. Файлы проекта хранят в репозитории, к которому имеют доступ вся команда проекта (рис.4). Как только разработчик напишет новый модуль или сделает рефакторинг, он сразу передает его в репозиторий и добавляет в проект. Чтобы проверить работоспособность проекта после внесения изменений, запускаются автотетсы на проверку интеграции и проводят регрессионное тестирование.

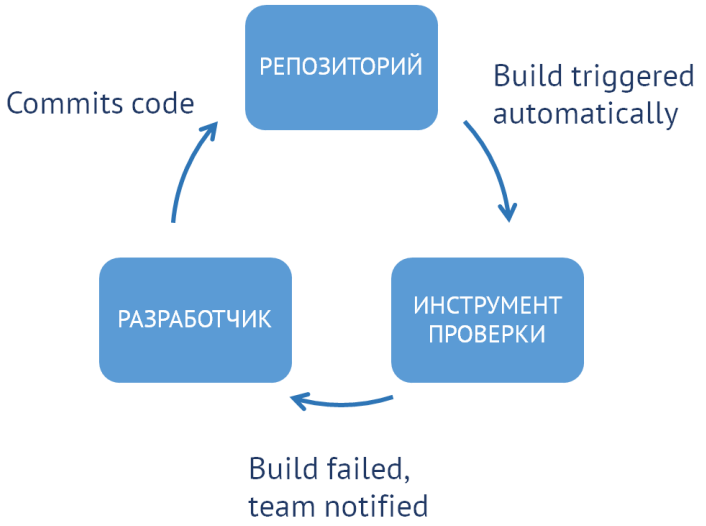

Рисунок 4 – Идея интеграции

При тестировании используют специальные программные инструменты, такие как Jenkins. Также в обеспечении непрерывной интеграции участвуют системы контроля версий – репозитории, например Git, в которых сохраняют файлы проекта и ветви проекта. Для облачного хранения данных проекта используют специальные платформы для хостинга и совместной работы, например Github или Gitlab.

Рисунок 5 – Сервисы по обеспечению непрерывной интеграции

Программные среды для разработки ПО

При проверке работоспособности программного продукта, тестировщики могут иметь доступ к программному коду. В этом случае тестирование проводят методом «Белого ящика».

Тестировщику необходимо открыть скрипт в принятой на проекте среде IDE, например Visual Studio Code.

IDE (Integrated Development Environment) и редактор кода – это виды программного обеспечения, предназначенные для работы над приложениями, их разработки и тестирования.

Состав IDE:

– текстовый редактор для написания и редактирования кода;
– компилятор – инструмент, позволяющий перевести текст, написанный на языке программирования, в набор машинных кодов;
– отладчик, проверяющий код и устраняющий в нем ошибки;
– инструментов для автоматизации сборки кода, ускоряющих процесс разработки.

Выбор IDE зависит от задач проекта, поэтому тестировщику полезно ознакомиться с возможностями Visual Studio Code, PyCharm, Notepad++, Sublima Text, Netbtans (рис.6).

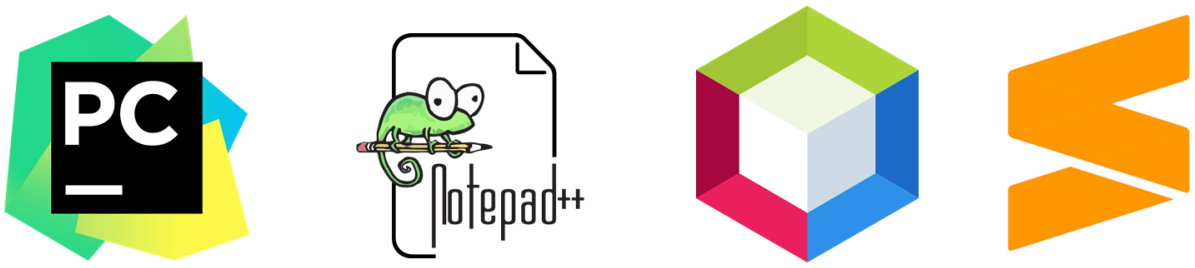

Рисунок 6 – Примеры программных сред и IDE

Системы управления тестированием

Все тест-кейсы, артефакты тестирования, сопроводительная документация могут быть сохранены в системе управлением тестированием.

Системы управления тестированием (Test Management System, TMS) используются для хранения информации о том, как должным образом проводить тестирование, осуществление очередности проведения тестирования в соответствии с его планом, а также для получения информации в виде отчетов о стадии тестирования и качестве тестируемого продукта.

К базовым задачам таких систем относят.

1. Создание и управление проектами.
2. Создание пользователей и проектных ролей для пользователей.
3. Удобная интеграция с автоматическими тестами.
4. Работа с тестовыми артефактами: тест-план, тест-кейс, чек-лист, общие шаги.
5. Версионирование тест-кейса/чек-листа.
6. Создание пользовательских атрибутов/конфигураций.
7. Понятная система отчётности.
8. Встроенная система баг-трекинга.
9. Возможность оповещения коллег внутри и вне системы.
10. Возможность интеграции с другими инструментами.

Примером отечественной системы управления тестированием служит Test It (https://testit.software/product). Также на проектах могут использовать облачную систему Qase (https://qase.io/).

Баг-трекинговые системы

Вместе с системами управления тестированием и системами управления проектами применяют баг-трекинговые системы.

Системы отслеживания ошибок (анг. bug tracking systems, далее – BTS) – прикладные программы, используемые разработчиками программного обеспечения с целью упрощения контроля над этапами разработки проекта и устранения возникающих ошибок

К таким системам относят многофункциональную платформу Test It, Яндекс.Трекер, Mantis.

Генераторы тестовых данные

При создании тестовый среды и тестового стенда необходимо использовать наборы тестовых данных.

– **Реальные тестовые данные**: требуются для проверки функциональности программной системы;

– Тестирование на **вымышленных тестовых данных** покажет, как отреагирует система на ввод фактически неправильных значений, будет ли выведено сообщение для пользователя о том, что таких данных не может быть.

– **Пограничные значения** выявят проблемы, которые часто встречаются именно на границах классов эквивалентности;

– Тестировщику следует проверить, как реагирует система на ввод информации в **неправильном формате**, есть ли данные об ошибках;

– Также важно протестировать продукт на использование **полей без ввода каких-либо значений**.

Наборы тестовых данных важны при тестировании бизнес-логики или базы данных. Для их создания используют генераторы, например, сервис datprof.com. Они позволяют анонимизировать реальные данные или сгенерировать с нуля синтетические (с характеристиками настоящих данных).

Вспомогательные программные инструменты

При документировании дефекта необходимо предоставлять информацию о том, где найден дефект и в чем он себя проявляет. Бывает так, что словами описать увиденное сложно. Поэтому тестировщики делают скриншот экрана или записывают видео. Примером такого полезного инструмента является FastStone.

Процесс реализации программного продукта сопровождает документация. Ее можно создавать в любом редакторе, который принят в организации, с помощью Markdown или Word, Latex или Notepad++ (рис.7).

Рисунок 7 – Программные инструменты для создания документации

В своей работе тестировщику необходимо взаимодействовать с разными программными средами, поэтому к важным в этой профессии навыкам относят умение быстрой адаптации и знакомства с теми программными средами, которые приняты на проекте.

Список источников

1. ГОСТ Р 56920-2016/ISO/IEC/IEEE 29119-1:2013. Системная и программная инженерия. Тестирование программного обеспечения. Часть 1. Понятия и определения
2. Орлов С.А. Программная инженерия. Технологии разработки программного обеспечения: учебник. — СПб: Питер, 2020. С.640.
3. Система управления проектами Kaiten. URL: https://kaiten.ru/
4. Авторство: Apache NetBeans. https://cwiki.apache.org/confluence/display/NETBEANS/Apache+NetBeans+Logo+Contest, Apache License 2.0,
5. Авторство: https://commons.wikimedia.org/w/index.php?curid=72889801
6. Авторство: Copyright (C) 2014 Uri Herrera and others, KDE Visual Design Group;. KDE github;
7. Авторство: Yandex LLC. https://connect.yandex.ru/ seen on the page, Общественное достояние https://commons.wikimedia.org/w/index.php?curid=73703922