Все современные достижения человечества делают нашу жизнь максимально простой. И сегодня все можно выполнить буквально одним нажатием клавиши. Но чтобы достичь подобного результата, необходимо создать максимально качественную и работоспособную логику, которая могла бы функционировать даже в самые критические моменты.

Конечно, со временем, созданные технологии могут функционировать не так корректно, как предполагалось изначально. Да и итоги применения того или иного ПО могут не соответствовать заявленным требованиям. Все это доказывает тот факт, что процесс тестирования любого программного обеспечения играет крайне важную роль.

Чтобы разобраться в интеграционном тестировании, сначала надо определить понятие традиционного процесса проверки ПО. Итак, ***тестирование программного обеспечения*** – это специфическая деятельность, которая проверяет, соблюдаются ли заявленные требования в текущей работе веб-продукта.

Другими словами, фактический результат работы ПО должен соответствовать ожидаемому результату. Иначе в запрограммированную структуру программного обеспечения придется вносить коррективы или переписывать код заново.

Проверка работоспособности ПО проводиться на разных уровнях. Базовыми среди них являются:

Уровни тестирования ПО

Согласно отображенной хронологии, процедура интеграционного тестирования проводиться сразу же после выполнения модульных тестов. Если проанализировать значение термина интеграция, то станет ясно, что процесс интеграционного тестирования выстраиваться на

основе проверки программных модулей, которые комбинируются в специальные группы. Именно из компонентов и складывается общее понятие программной системы.

Любая программная система тестируется как единое целое, а такой процесс как раз и называется ***интеграционным тестированием***. Его главной **задачей является проверка разных модулей системы при их системном объединении.**

Интеграционное тестирование входит в состав тестирования белого и черного ящика. В сегодняшних реалиях большинство компаний как раз предпочитают выполнять проверки на основе модульных и функциональных тестов.

ВИДЫ ПОДХОДОВ К ИНТЕГРАЦИОННОМУ ТЕСТИРОВАНИЮ

Есть сразу 4 основных типа и подхода к процессу интеграционного тестирования, которые стоит рассмотреть более детально.

Типы интеграционного тестирования:

- **Большой взрыв** (англ. Big bang approach);
- **Снизу вверх** (англ. Bottom-Up Approach);
- **Сверху вниз** (англ. Top-Down Approach);
- **Смешанный / сэндвич** (англ. Hybrid / Sandwich).

БОЛЬШОЙ ВЗРЫВ (АНГЛ. BIG BANG APPROACH)

Процесс разработки ПО предполагает, что созданные и запрограммированные модули и системные компоненты соединены между собой. При объединении эти модули тестируются как единое целое. После проведения юнит-тестов, модули также проверяются вместе, еще до образования целостной программной системы.

Может возникнуть вопрос: **в чем разница между тестированием ПО в целом и интеграционным тестированием модулей?** Все просто! Главное отличие в том, что при интеграционном тестировании работы проводятся исключительно с отдельными модулями, а при проверке всего ПО тестируются все системные параметры.

Представленная ниже диаграмма красноречиво демонстрирует, что именно означает метод «большого взрыва» в процессе проведения интеграционных проверок.

Метод большого взрыва

К слову, подобный подход имеет несколько преимуществ и даже недостатков.

Преимущества:

1. Весьма удобен в использовании при тестировании небольших систем.
2. Быстрое нахождение ошибок, а значит, существенная экономия время, которое может быть потрачено на разработку и доработку уже используемого функционала.

Недостатки:

1. Так как модули завязаны на одной системе, порой очень трудно найти источник дефектов.
2. Если в системе используется много модулей, может уйти достаточно времени, чтобы пересмотреть все реализованные функциональности.

СНИЗУ ВВЕРХ (АНГЛ. BOTTOM-UP APPROACH)

Подобный подход подразумевает проверку низкоуровневых систем для начала: вместе и по отдельности. Другими словами процесс тестирования начинается с внутреннего уровня и постепенно доходит до наиболее критичных позиций.

Представленная далее схема красноречиво свидетельствует о том, что модули верхнего ранга не могут быть интегрированы в ПО до тех пор, пока не будет завершено тестирование модулей нижнего порядка.

Подход снизу вверх

При подходе «снизу вверх» может использоваться ***Драйвер***, который выступает в роли специального «соединителя» между модулями нижнего и верхнего уровней.

Преимущества:

1. Создание отдельных модулей может совершаться при применении метода интеграционного тестирования по схеме «снизу вверх», так как тестирование самых критических моментов начинается с тестов модулей нижнего порядка.
2. Если определенный модуль перестает функционировать, его ошибка может быть сразу же исправлена.
3. Требуется минимальное время на идентификацию и устранение ошибок.

Недостатки:

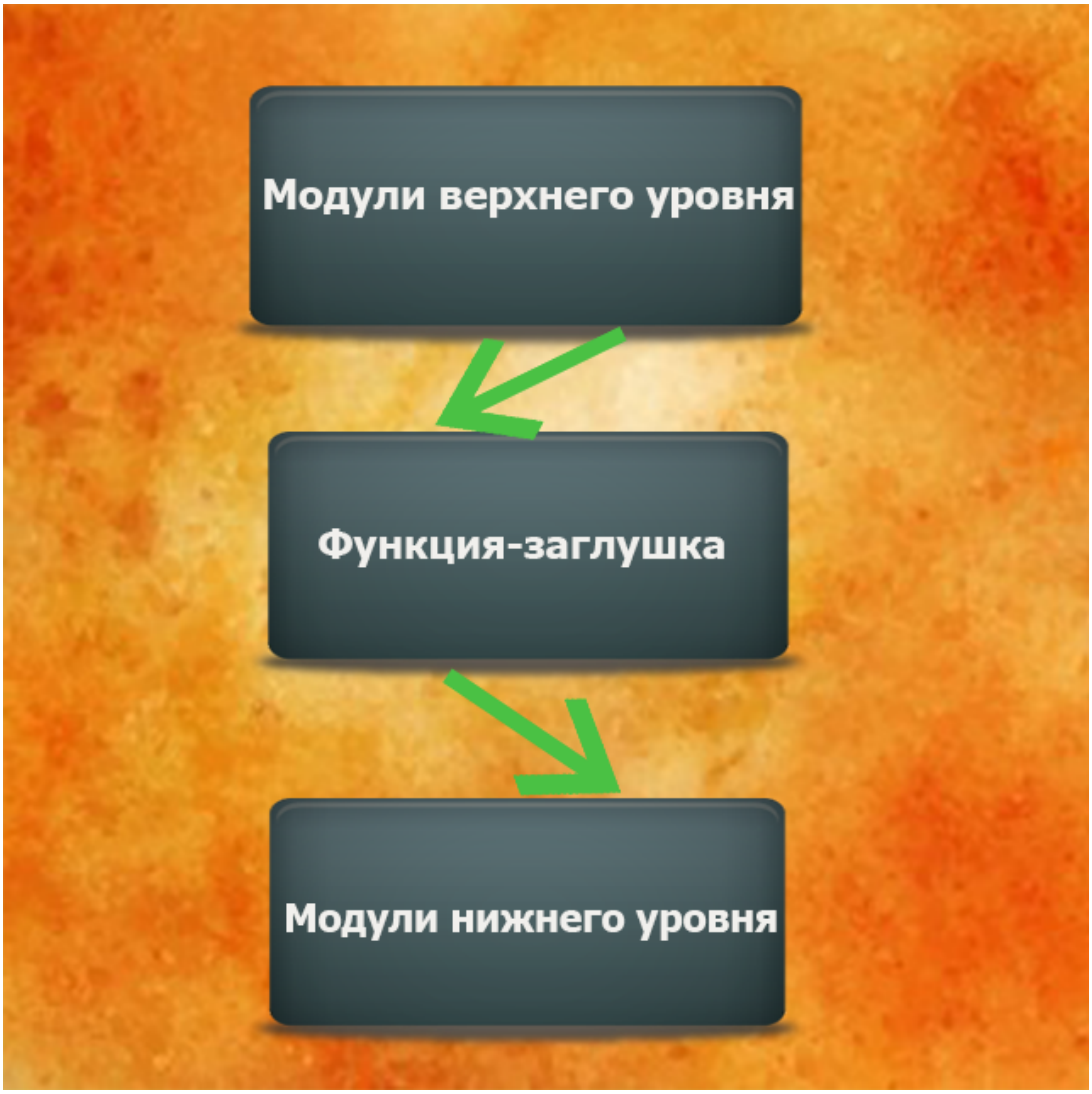
1. Общее время проверки всех модулей довольно долгое, так как продукт не может быть представлен в релиз пока не пройдет тестирование от самого нижнего до самого верхнего модулей.
2. Если ПО содержит много небольших модулей мелкого уровня, которые очень сложны в своей имплементации, то для завершения процесса тестирования может потребоваться больше времени.

СВЕРХУ ВНИЗ (АНГЛ. TOP-DOWN APPROACH)

Это полная противоположность вышеописанной методике. Суть подхода «сверху вниз» заключается в первостепенном тестировании всех верхних модулей, и только затем QA специалист может приступать к проверке работоспособности нижестоящих модулей.

Большинство модулей нижнего уровня тестируются по отдельности, а затем выполняются проверки в совокупности реализованных модулей. По аналогии с подходом «снизу вверх», данный метод также зависим от вызова специальной связующей функции под названием ***«Функция-заглушка»*** (англ. Stubs).

Заглушки – это специальные логические операторы с коротким программным кодом, которые применяются для приема входных данных нижними модулями от модулей верхнего уровня при интеграционном тестировании.



Подход сверху вниз

Преимущества:

1. Легко обнаружить неисправности или ошибки в работе системы.
2. В первую очередь проверяются важные модули, а лишь потом модули нижнего порядка.
3. По сравнению с другими подходами, время на тестирование интеграции очень коротко.

Недостатки:

1. Если в модули нижнего уровня заложена важная логика, она не может быть протестирована в первую очередь, пока не завершится работа над проверкой верхних порядков.
2. Использование «заглушек»становится обязательным на всех последующих проектах.

СМЕШАННЫЙ / СЭНДВИЧ (АНГЛ. HYBRID/SANDWICH APPROACH)

Данный подход еще принято именовать как ***тестирование смешанной интеграции***.

Логики подходов «сверху вниз» и «снизу вверх» максимально объединены в этот подход. А значит, его запросто можно считать смешанным, своего рода гибридным методом в интеграционном тестировании.

Итак, самый верхний модуль тестируется отдельно, при этом модули нижнего уровня интегрируются и проверяются с модулями верхнего уровня.

Преимущество:

- Идеально подходит для больших проектов, работа над которыми длится очень долгое время.

Недостаток:

- Цена подобного тестирования очень высока, так как данный подход включает в себя сразу несколько модулей проведения интеграционного тестирования.

БОЛЬШИЕ ПРЕИМУЩЕСТВА ИНТЕГРАЦИОННОГО ТЕСТИРОВАНИЯ

Использование методики одновременного тестирования для разных модулей – это очень удобно, практично и не слишком сильно затратно в финансовом плане, по сравнению с другими типами тестов.

Интеграционные проверки могут использоваться на любой стадии разработки и тестирования ПО. По сравнению с другими видами тестов, данные проверки могут охватывать любые объемы программного кода за один спринт, ведь есть так называемые подходы «снизу вверх» и «сверху вниз».

Бесспорно, интеграционное тестирование крайне важно использовать особенно на тех проектах, где требования разработки весьма переменчивы и структура создаваемой логики может неоднократно подвергаться корректировкам и пересмотрам.

ЗАЧЕМ ИСПОЛЬЗОВАТЬ ИНТЕГРАЦИОННОЕ ТЕСТИРОВАНИЕ?

Люди из сферы IT-индустрии прекрасно знают, насколько переменчивой может быть рабочая атмосфера на проектах. Каждый день ранее утвержденные требования могут пересматриваться, редактироваться и банально терять свою актуальность.

Все это приводит к созданию большого количества строчек программного кода, который в любом случае необходимо тщательно тестировать на предъявленные заранее требования.

Идеальным инструментом для подобных целей как раз и выступает интеграционное тестирование, позволяющее классифицировать программный код на блоки (модули). Интеграция проверки ПО очень важна, так как в релиз должен поступать исключительно работоспособный и качественный продукт, ликвидность которого в своей нише будет максимальной (так следует в теории).

ШАБЛОННЫЕ ПРИМЕРЫ ИНТЕГРАЦИОННОГО ТЕСТИРОВАНИЯ

Для эффективной визуализации примера представим, что мы работаем над ПО для личного кабинета сотрудников внутри определенной корпорации.

Итак, наша программа будет содержать такие структурные модули:

- Логин работника;
- Отчет о работе сотрудников;
- Страница заработной платы и уровень отчисления налогов.

В процессе первоначальной работы над продуктом мы создаем логику регистрации и авторизации сотрудника (страницы с формами для ввода логина и пароля). При правильной регистрации и верификации, клиент должен перенаправляться системой на страницу личного кабинета (страница персональной отчетности).

А если после ввода информации, сотрудник не переходит на страницу с личными данными, значит, в системе есть ошибка. И чтобы найти и в последующем исправить эту неточность созданной логики, и используется интеграционное тестирование.

Вторым примером интеграционного тестирования может послужить такая ситуация:

Пользователь каждый день проверяет свою электронную почту. Все почтовые клиенты должны предоставлять собой один сформированный набор функциональных возможностей ***(ЛК-входящие/исходящие письма-папка спам-выход из аккаунта).***

Итак, если в нашем распоряжении есть сервера почтовых агентов, мы может начать с простых модульных тестов. Далее, после совпадения учетных данных, процесс верификации на странице входа должен перенаправить нас на страницу входящих сообщений. Если этого не происходит, значит необходимо использовать интеграционные тесты чтобы более конкретно понять в чем же проблема.

ИТОГИ

В современном ИТ-мире, надобность в постоянном проведении интеграционного тестирования становится все более популярной и востребованной. Учитывая модульную гибкость интеграций, подобный вид тестирования можно применять как на больших, так и мелких проектах с различной направленностью.

Ежедневная практика использования интеграционного тестирования позволяет клиентам добиваться максимального значения коммерческой выгоды для своего веб-продукта.