УДК 004.05

РАЗРАБОТКА ФРЕЙМВОРКА ДЛЯ АВТОМАТИЗАЦИИ ИНТЕГРАЦИОННОГО ТЕСТИРОВАНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ ГОСУДАРСТВЕННЫХ ЗАКУПОК С ФЕДЕРАЛЬНОЙ ЭЛЕКТРОННОЙ ТОРГОВОЙ ПЛОЩАДКОЙ

Чжао М.Н., магистрант, МОиАИС БИТ, 2 курс
Научный руководитель: Савкин Д.А., доцент
Балтийский федеральный университет имени Иммануила Канта
г. Калининград

В условиях постоянного развития цифровых технологий и повсеместной цифровизации ключевых секторов экономики, включая такие секторы как: банковский, государственных закупок, коммерции. Торги по муниципальным и федеральным заказам стали электронными, а торговые площадки превратились в сложные многокомпонентные экосистемы, упрощающие и организующие данный процесс. В виду этого они имеют множество интеграций с различными внешними системами от информационных до платежных. Примером такой электронной торговой площадки является (ЭТП) «Сбербанк — Автоматизированная система торгов» («Сбербанк-АСТ»). Данная ЭТП является частью экосистемы Сбербанка для проведения торгов согласно контрактной системе заложенной в законодательстве Российской Федерации [1]. Торговая площадка интегрируется с информационной системой (ИС) «Автоматизированный центр контроля — Государственный заказ» («АЦК-Госзаказ») предназначенной для автоматизации исполнения процесса государственных (муниципальных) закупок. В свою очередь автоматизированный центр контроля интегрируется с десятками государственных реестров, платежными шлюзами, аналитическими платформами. Такая многоуровневая архитектура, с одной стороны, обеспечивает высокую функциональность и помогает упростить закупочный процесс за счет автоматизации различных этапов государственных закупок, а с другой — увеличивает риски сбоев на стыке взаимодействующих систем. Подобные сбои могут привести к нарушению процессов государственных закупок, финансовым потерям, а также репутационным и юридическим рискам. В связи с этим интеграционное тестирование выступает неотъемлемым этапом разработки и сопровождения подобных комплексных решений. Данный вид тестирования позволяет обеспечить проверку корректности взаимодействия интегрируемых систем в условиях пользовательских сценариев использования.

В свою очередь автоматизированный подход к интеграционному тестированию позволяет не только проверить критически важные интеграционные функции, но и предоставляет ряд преимуществ по сравнению

с ручным подходом. К основным преимуществам автоматизации интеграционного тестирования можно отнести:

- Ускорение тестирования. Один и тот же тестовый сценарий вручную выполняется дольше чем автоматизировано. Это обусловлено тем, что действия сценария выполняются быстрее, а также тем, что автотесты (автоматизированные тесты) поддерживают параллельное выполнение [2]. Данный фактор особенно важен в таком ресурсоемком виде тестирования, где от специалистов по тестированию требуется взаимодействовать с различными интерфейсами тестируемых систем;
- Повторяемость. Автоматизированный тест пишется один раз, а далее может быть выполнен множество раз;
- Увеличение частоты тестирования. Ускорение тестирования и повторяемость позволяют выполнять цикл тестирования продукта чаще;
- Минимизация влияния человеческого фактора на результаты тестирования. В рамках автотеста выполняются только те действия, которые заложены специалистом по автоматизации тестирования при написании, что исключает возможность совершения ошибки при многократном выполнении в отличии от ручного подхода;
- Автоматический сбор отчетности. При выполнении автотестов отчетность о результатах собирается автоматически;
- Уменьшение издержек на выпуск новой версии тестируемого продукта. За счет того, что тестирование выполняется быстрее и уменьшается количество специалистов, требуемых на его выполнение, происходит уменьшение издержек.

Разработка специализированного фреймворка для автоматизации интеграционного тестирования ЭТП «Сбербанк-АСТ» и ИС «АЦК-Госзаказ» является главной задачей в рамках автоматизации данного вида тестирования, поскольку специализированный фреймворк решает ряд ключевых проблем, включая такие как:

- Создание единой структурированной среды для автоматизированных интеграционных тестов. Благодаря этому все автотесты пишутся по единым правилам и в едином стиле;
- Ускорение написания новых автотестов за счет переиспользования программного кода фреймворка в различных тестах на его базе;
- Упрощение поддержки автоматизированных тестов. Поскольку программный код фреймворка преиспользуется между автотестами, то в случае необходимости изменения поведения теста необходимо изменить лишь общий переиспользуемый метод или группу методов;
- Интеграция в линии поставки в различных CI/CD системах [3]. Ввиду того, что фреймворк организует выполнение всех

автоматизированных тестов, выполняемых без участия человека, то он может быть интегрирован в линию поставки программного продукта в роли одного из этапов линии.

Первоочередной задачей при реализации фреймворка является выбор языка программирования, на котором будет реализован фреймворк. Поскольку ИС «АЦК-Госзаказ», в линию поставки которой планируется интегрировать фреймворк, разработана на языке программирования (ЯП) Java, и поскольку данный язык широко применяется для автоматизации тестирования, а также имеет обширный инструментарий интегрируемых библиотек для тестирования, то было принято решение разрабатывать фреймворк и автоматизированные тесты на данном ЯП. Благодаря такому выбору обеспечивается высокая интеграция автоматизированного тестирования в процесс разработки информационной системы.

После выбора языка программирования для разработки фреймворка необходимо определить перечень библиотек и инструментов, которые потребуются для реализации проекта. Основываясь на том, что в рамках сценариев, которые необходимо автоматизировать, выполняется взаимодействие с различными интерфейсами: пользовательским, прикладным программным, а кроме того, с базой данных «АЦК-Госзаказ», а также на том, что автотесты будут интегрированы в линию поставки данной ИС, то были выбраны следующие инструменты:

- Клиентская библиотека Selenium WebDriver — для взаимодействия с пользовательским интерфейсом тестируемых систем;
- HTTP-клиент OkHttp — для обращения к прикладным программным интерфейсам тестируемых приложений;
- Драйвер для базы данных PostgreSQL на основе технологии JDBC [4] — для выполнения запросов к базе данных «АЦК-Госзаказ»;
- Библиотека управления автоматизированными тестами TestNG — для конфигурирования и запуска набора тестов;
- Библиотека построения отчетности Allure Report — для автоматизации процесса сбора отчетности о выполненных тестах;
- Система автоматической сборки Apache Maven — для сборки проекта и управления выбранными зависимостями.

Для реализации архитектуры фреймворка были использованы рекомендации международного квалификационного совета по тестированию программного обеспечения (ISTQB). В рамках международного совета выполняется агрегация знаний по различным аспектам тестирования. Для построения специализированных фреймворков данной организацией была выработана обобщенная архитектура подобных решений названная Generic Test Automation Architecture (gTAA) [5] представленная на рисунке 1.

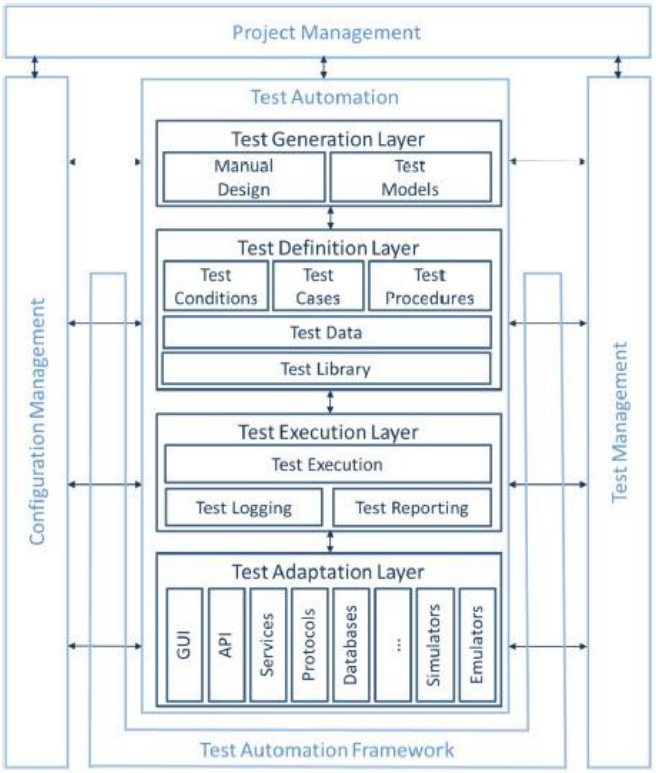

Рис. 1 ISTQB Generic Test Automation Architecture

В данной архитектуре описываются распространенные компоненты фреймворков и их взаимодействие, а также окружение, с которым может взаимодействовать фреймворк. По gTAA фреймворк разделяется на слои, а слои на модули. Всего в представленной обобщенной архитектуре представлено четыре слоя:

- Test Adaptation Layer (слой адаптеров). На данном слое реализуются модули позволяющие взаимодействовать с интерфейсами тестируемой системы или систем на основе различных драйверов и иных инструментов;
- Test Execution Layer (слой выполнения). В пределах данного слоя реализуются модули, отвечающие за отчетность и выполнение тестов;
- Test Definition Layer (слой определения). На слое определения создаются модули, определяющие тестовые действия, переиспользуемые в автоматизированных тестах, данные для тестов, а также реализуется конфигурация запуска тестов;
- Test Generation Layer (слой генерации). Слой генерации включает в себя общие правила реализации непосредственно самих автотестов, то есть их язык;

Данная архитектура была взята за основу для реализации фреймворка. На основании ранее выбранных инструменов и особенностей тестируемых систем, была реализована следующая структура проекта фреймворка, представленная на рисунке 2.

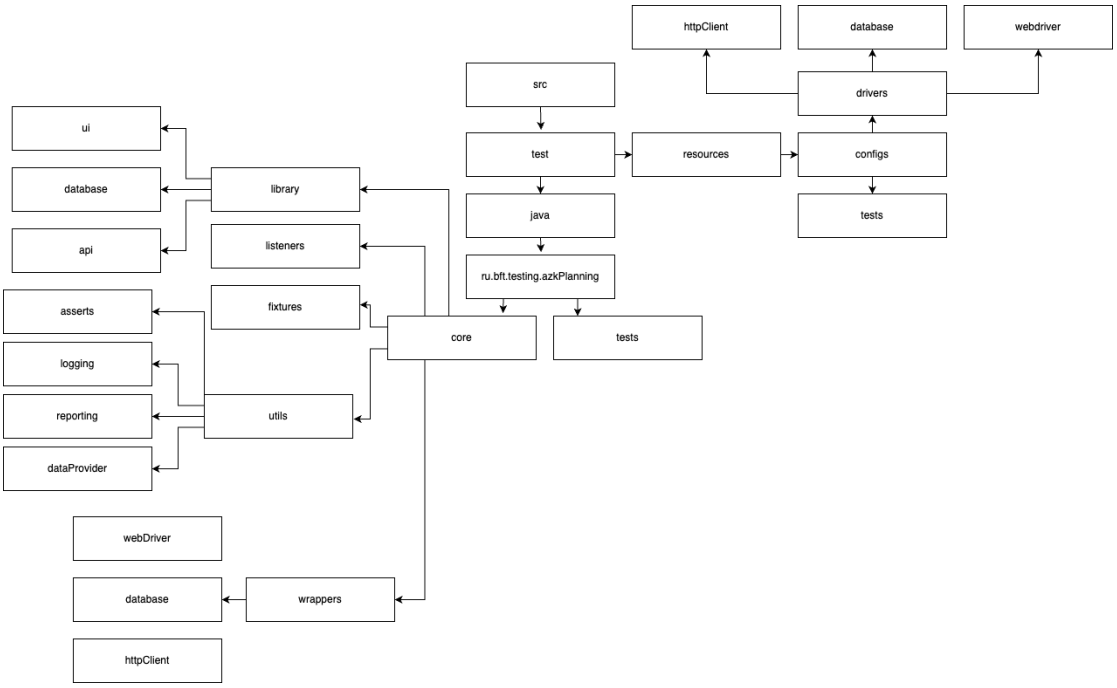

Рис. 2 Структура проекта фреймворка автоматизированного интеграционного тестирования ЭТП «Сбербанк-АСТ» и ИС «АЦК-Госзаказ»

В пакете wrappers реализован слой адаптеров и модули адаптеров для каждого из используемых в тестах интерфейсов тестируемых систем. В пакетах utils, fixtures, listeners реализованы модули слоя выполнения, отвечающие за логирование, отчетность, фикстуры, а также выполнение автоматизированных тестов. В пределах пакета library реализованы модули, в которых описываются переиспользуемые в автотестах шаги. Данные модули относятся к слою определения. Все конфигурационные файлы фреймворка помещены в директории resources. В пакете tests располагаются автотесты.

Автоматизированные тесты на базе данного фреймворка представляют из себя метод, не возвращающий ничего, помеченный аннотацией @Test [6]. Для демонстрации возможностей фреймворка был выбран тестовый сценарий для проверки получения сообщения типа извещение из «Сбербанк-АСТ» в очередь сообщений «АЦК-Госзаказ». Сценарий представлен в таблице 1.

Табл. 1 Сценарий «Проверка получения сообщения типа извещение из ЭТП "Сбербанк-АСТ" в ИС "АЦК-Госзаказ"»

Сценарий:	Выполнение логина с неверными данными	
Шаг сценария	Данные для ввода	Ожидаемый результат
Имитировать отправку сообщения типа извещение из ЭТП «Сбербанк-АСТ»	Сообщение типа извещение в формате XML	От ИС «АЦК-Госзаказ» получено подтверждение получения сообщения
Проверить очередь сообщений типа извещение в ИС «АЦК-Госзаказ»	-	Сообщение отображено в очереди сообщений ИС «АЦК-Госзаказ»

Продолжение табл. 1

Тело сообщения корректно обработано и отображается в ИС «АЦК-Госзаказ»	-	В пользовательском интерфейсе отображается полученное тело сообщения

Приведенный сценарий был реализован на базе фреймворка и представлен на рисунке 3.

```
@Test(description = "Проверка получения сообщения типа извещение из ЭТП 'Сбербанк-АСТ' в ИС 'АЦК-Госзаказ'")  new *
public void checkMessageNotification() {
    sberAst.mock.messages.send( type: "notification",  message: "notification_01.xml");
    azkgz.ui.queue.check( queueName: "OOS_DOC_3215",  messageToFind: "notification_01.xml");
    azkgz.ui.queue.validateAttributes( queueName: "OOS_DOC_3215",  messageToFind: "notification_01.xml");
}
```

Рис. 3 Реализация сценария «Проверка получения сообщения типа извещение из ЭТП "Сбербанк-АСТ" в ИС "АЦК-Госзаказ"» на базе разработанного фреймворка

В результате был разработан фреймворк для автоматизированного интеграционного тестирования электронной торговой площадки «Сбербанк — Автоматизированная система торгов» и информационной системы «Автоматизированный центр контроля — Государственный заказ» на языке программирования Java с применением шаблона рекомендаций международного квалификационного совета по тестированию программного обеспечения.

Список литературы:

1. Минфин России, Контрактная система – URL: https://minfin.gov.ru/ru/perfomance/contracts/. Текст: электронный.
2. TestNG Documentation, 3.11. Parallelism and time-outs – URL: https://testng.org/#_parallelism_and_time_outs. Текст: электронный.
3. Red Hat, What is CI/CD? – URL: https://www.redhat.com/en/topics/devops/what-is-ci-cd. Текст: электронный.
4. Oracle Help Center, JDBC Introduction – URL: https://docs.oracle.com/javase/tutorial/jdbc/overview/index.html. Текст: электронный.
5. Sogeti Labs, BENEFITS OF A GENERIC TEST AUTOMATION ARCHITECTURE – URL: https://labs.sogeti.com/benefits-of-generic-test-automation-architecture/. Текст: электронный.
6. TestNG Documentation, 5 - Test methods, Test classes and Test groups – URL: https://testng.org/doc/documentation-main.html. Текст: электронный.