Министерство образования Республики Беларусь
Учреждение образования
Белорусский государственный университет
информатики и радиоэлектроники

УДК 004.415.53

Улащик
Алексей Николаевич

Технологии автоматизации тестирования на основе моделирования распределенного выполнения тестов

АВТОРЕФЕРАТ
на соискание степени магистра
информатики и вычислительной техники
по специальности 1-40 81 01
«Информатика и технологии разработки программного обеспечения»

Научный руководитель
Новиков Владимир Иванович
кандидат технических наук

Минск, 2015

Процесс тестирования программного обеспечения (ПО) представляет собой столь же неотъемлемую часть его разработки, как и проектирования. Роль тестирования еще более возрастает с применением прогрессивной итеративной и инкрементальной методологии разработки ПО.

Задачами современного тестирования являются не только обнаружение ошибок в программах, но и выявление причин их возникновения. Такой подход позволяет разработчикам программного обеспечения действовать максимально эффективно, быстро устраняя возникающие ошибки.

Понимание важности процесса тестирования приводит к возникновению тенденций, направленных на применение промышленных способов проверки качества программного обеспечения. Наиболее важным направлением в данной области является внедрение различных систем автоматизированного тестирования. В результате работы автоматических тестов мы получаем понятный и информативный отчет об их выполнении, содержащий все этапы работы, и дефекты, обнаруженные в ходе тестирования программного продукта.

Целью данной магистерской работы является разработка эффективных технологий автоматизированного тестирования, рассмотрение возможности использования современных, сверхвысокоуровневых языков программирования на примере языка Python, а также применение основных подходов и концепций к тестированию современных программных продуктов, используя данный язык. Python- это мультипарадигменный язык, поддерживающий как функциональный, так и объектно-ориентированный подходы.

Задачей данной магистерской работы является написание фреймворка для автоматизированного тестирования классического веб-приложения с использованием языка программирования Python и также построение модели распределенного выполнения тестов, который позволяет быстро провести тестирование, максимально покрывая функциональность продукта. Рассматриваемая система тестирования - модульная, что позволяет гораздо быстрее реализовать новые тесты и легко поддерживать уже существующие.

В работе рассмотрен язык программирования Python как основное средство для построения тестового фреймворка, т.к. он имеет чёткий и последовательный синтаксис с продуманной модульностью и масштабируемостью, благодаря чему исходный код, написанный на Python, легко читаем и имеет небольшой объем; Python поддерживает как объектно-ориентированный подход к программированию, так и функциональный; Python поддерживает динамическую типизацию, то есть тип переменной определяется только во время исполнения, что значительно уменьшает количество кода за счет отсутствия излишнего описания структур данных;

интерпретатор Python имеет интерактивный режим работы, при котором введённые с клавиатуры операторы сразу же выполняются, а также Python поставляется в комплекте с большим количеством библиотек для выполнения множества операций, таких как логирование, работа в Web и т.д. Поэтому использование языка Python позволяет применить основные подходы и концепции к тестированию современных программных продуктов.

В магистерской работе рассмотрена практическая реализация основных концепций автоматизированного тестирования. В результате практических исследований создан прототип тестового фреймворка, позволяющий проводить функциональное тестирование программного обеспечения по заранее созданным тестовым сценариям. За счет применения объектно-ориентированного подхода в тестовом фреймворке заложена архитектура, позволяющая в дальнейшем расширять функциональные возможности, не изменяя уже существующий программный код. С помощью библиотек Selenium и Requests созданы прототипы для тестирования графических интерфейсов, а также веб-ориентированных сервисов. Такой подход сочетает в себе как новизну в рамках использования языка программирования Python, так и лучшие мировые практики в создании фреймворков для автоматизированного тестирования.

Анализ полученного программного кода позволяет однозначно говорить о более высокой эффективности использования сверхвысокоуровневых языков программирования, таких как Python, в автоматическом тестировании. За счет использования синтаксических особенностей языка программирования Python, а также благодаря встроенной поддержке системы интроспекции и механизмов управления данными, таких как дескрипторы, полученный код значительно меньше аналогичной реализации на Java или C#.

Разработанный фреймворк и результаты работы могут быть внедрены и использованы при проектировании современных систем автоматизированного тестирования.

Большинство программных продуктов, выпускаемых в настоящее время, являются веб-ориентированными приложениями, рассчитанными на работу в интернет-браузере. В эпоху высокой интерактивности и взаимодействия в процессе разработки программ, когда многие организации используют методологию Agile в той или иной форме, автоматизация тестирования становится необходимостью.

Автоматизированное тестирование обладает множеством достоинств, связанных, главным образом, с высокой скоростью выполнения тестов и возможностью выполнять однотипные тесты снова и снова. Существует

большое количество как коммерческих, так и бесплатных инструментов, помогающих в разработке автоматизированных тестов.

В современном мире информационных технологий сложность программных комплексов увеличивается каждый день, растет количество функциональных возможностей и технологий, используемых в программных продуктах. Использование большого количества компонентов и сторонних систем ведет к увеличению количества ошибок и падению качества программных продуктов. Как правило, на больших проектах невозможно обойтись без автоматизации тестирования для поддержания высокого качества программного продукта. В связи с этим встает задача разработки программных решений для обеспечения автоматизации тестирования.

Основной задачей тестового фреймворка является обеспечение интерфейса для проведения тестов над системой. Фреймворк является сложной многомодульной системой и его разработка схожа с разработкой программного комплекса.

Для обеспечения высокого качества тестового покрытия необходимо наличие функциональных тестов для Web-application части приложения, а также зачастую необходимо протестировать WebServices, которые предоставляют собой программный продукт. Эти два вида тестирования покрывают около 80% функциональности всех создаваемых сейчас приложений.

Ожидаемая структура типичного тестового фреймворка включает в себя следующие модули: модуль для работы в Web; модуль для работы с WebServices; тестовый модуль для запуска тестов, контроля их выполнения и предоставления отчетной информации. Возможно также наличие дополнительных модулей, например таких, как модуль прямого доступа к базе данных для дальнейшей валидации, или модулей доступа к сторонних сервисам для обработки результатов тестирования.

Как правило, фреймворки, разработанные в соответствии с данной концепцией, строятся на базе opensource-решений. Однако для их построения требуется привлечение высококвалифицированных специалистов в данной области. Основным преимуществом данного подхода является свобода реализации необходимого поведения, гибкость конечного решения и возможность масштабирования продукта.

Основной сложностью при разработке тестового фреймворка является необходимость построения масштабируемой и, одновременно, модульной системы с возможностью быстрого расширения.

Анализ существующих систем тестирования на базе Java, C# и других языков программирования показывает, что использующиеся системы сложны, а также тяжелы в обслуживании исходя из особенностей

используемой платформы. Необходим поиск решений на других языках программирования, которые обеспечат идентичный функционал, но облегчат процесс создания и обслуживания кода. Задача магистерской работы- построение тестового фреймворка на основе языка программирования Python. Функциональные и объектно-ориентированные возможности, а также легкий в понимании и обучении синтаксис этого языка делают его наиболее подходящим к использованию в рамках автоматизированного тестирования. Такая система будет позволять проводить тестирование как графического интерфейса, так и Web- ориентированных сервисов согласно классической концепции построения тестовых фреймворков.

Python – мультипарадигменный язык программирования сверхвысокого уровня с динамической типизацией, автоматическим управлением памятью и удобными высокоуровневыми структурами данных, такими как словари, хэш-таблицы, списки и кортежи. Большая часть реализаций данного языка – интерпретаторы, т.е. написанный код не проходит стадию предварительной компиляции. Python поддерживает классы, модули, пакеты (для структурирования программного обеспечения), обеспечивает обработку исключений, а также выполняет многопоточные вычисления. Язык программирования Python обладает простым и выразительным синтаксисом и поддерживает несколько парадигм программирования: объектно-ориентированное; функциональное; структурное; аспектно-ориентированное.

Python портируем и работает почти на всех известных платформах – от КПК до мейнфремов. Существуют порты под MicrosoftWindows, все варианты UNIX, MacOS, PalmOS, Symbian.

Несомненным достоинством является то, что интерпретатор Python реализован практически на всех платформах и операционных системах. Python имеет в наличии большое число подключаемых к программе модулей, обеспечивающих различные дополнительные возможности. Такие модули пишутся на C и на самом Python и могут быть разработаны достаточно квалифицированными программистами. Язык поддерживает такие конструкции, как замыкания, лямбда-выражения, списковые включения, декораторы и дескрипторы, позволяющие значительно сокращать объем исходного кода. Названные достоинства являются ключевым фактором для использования этого языка программирования в тестировании.

Для автоматизированного тестирования существуют следующие библиотеки: Selenium – библиотека для эмуляции пользователя в веб-браузера, позволяющая организовать тестирование веб- интерфейсов; Requests – библиотека для быстрого и семантически легкого создания запросов к веб- сервисам.

Существует большое количество библиотек для проведения юнит тестов в Python, однако в контексте данной работы рассмотрена собственная реализация для более детального рассмотрения процессов взаимодействия модулей.

Первым этапом выполнения тестов является сбор информации о тестовых случаях и формирование списка будущих тестов. Как правило, в промышленных случаях принято два вида получения данных: автоматический анализ файловой структуры с поиском тестов по какому-либо критерию, либо получение тестовой конфигурации в виде файла формата XML, JSONили YMAL.

В магистерской работе основной упор сделан на масштабируемость и легкую конфигурируемость, рассмотрен вариант получения входных данных в виде файла. Для простоты понимания используется XML как наиболее гибкий и широко распространенный формат.

Необходимо отметить, что в отличие от юнит- тестирования, при проведении автоматизированного тестирования на базе сценариев имеет место приоритет запуска того или иного тестового шага, а также зависимости одного шага от другого.

Ядро должно обеспечивать стабильную работу, несмотря на результаты тестирования, «отлавливать» и «гасить» все внешние исключения и выполнять тесты в соответствии с составленным планом. Неотъемлемой частью ядра любого тестового фреймворка является система построения отчетных данных. Традиционно существует большое количество форматов для предоставления отчетной информации: TXT, HTML, PDF и так далее. В работе отчет собран в XML файл. Данный формат достаточно хорошо читаем, а также хорошо конвертируем в любой другой удобочитаемый формат.

Для автоматизации тестирования веб-приложений рассмотрен Selenium в качестве основного инструмента. Selenium предоставляет несколько вариантов для идентификации элементов интерфейса, сравнения ожидаемого и наблюдаемого поведения тестируемого приложения. Одной из ключевых особенностей Selenium является возможность запуска одних и тех же тестов в различных браузерах. Selenium 2.0 содержит WebDriver API - объектно-ориентированный API, поддерживающий большее количество браузеров и лучше решающий проблемы тестирования современных веб-приложений.

Веб-сервисы (веб-службы) на сегодня являются неотъемлемым компонентом любого веб- ориентированного приложения. Они представляют собой распределенные компоненты приложений, доступные извне. Их можно использовать для интеграции компьютерных приложений, написанных на

различных языках программирования и выполняемых на различных платформах. Веб-службы не зависят от языка и платформы, так как между поставщиками существует договоренность об общих стандартах веб-служб. Модели веб-сервисов разделяются на две основные категории:

- REST - REpresentational State Transfer (передача состояния представления) - это новый способ создания и взаимодействия с веб-службами. В REST ресурсы имеют идентификаторы URI и управление ими происходит через операции с заголовками HTTP;
- SOAP/WSDL. В стандартных моделях веб-служб интерфейсы веб-служб предоставляются с помощью документов WSDL (тип XML) с URL-адресами. Последующий обмен сообщениями осуществляется через SOAP, другой тип документа XML.

Так как в подавляющем большинстве приложений на сегодняшний день используют REST архитектуру, в данной работе рассмотрено тестирование данного типа сервисов.

Веб-службы на основе REST ("RESTful") представляют собой коллекцию веб-ресурсов, идентифицируемых по своим URI. Каждый документ и каждый процесс смоделирован как веб-ресурс с уникальным идентификатором URI. Этими веб-ресурсами можно управлять с помощью действий, указанных в заголовке HTTP. Стандарты SOAP, WSDL и WS-* не используются. Вместо этого обмен сообщениями может быть проведен в любом формате – XML, JSON, HTML и т.д. Во многих случаях клиентом может служить веб-браузер.

Протоколом в REST является HTTP. Доступны только четыре метода: GET, PUT, POST и DELETE. Для запросов можно создавать вкладки, а ответы могут кэшироваться. Администратор сети может отслеживать работу службы RESTful путем просмотра заголовков HTTP.

REST является подходящей технологией для создания тех приложений, которые не требуют защиты, помимо той, которая доступна в инфраструктуре HTTP, и которым подходит протокол HTTP. Службы REST предоставляют весьма сложные функциональные возможности. Веб-службы RESTful применяются такими компаниями, как Flickr, GoogleMaps и Amazon.

Полученная модель предусматривает минимальный, но наиболее часто используемый функционал, необходимый при тестировании, а также рассматривает возможности подачи тестовых данных, организацию тестовых пакетов, концепцию построения системы для работы с графическими интерфейсами и удаленными веб-сервисами. Данные наработки создают предпосылки для практической реализации, которая будет описана в следующей главе.

Для разработки тестового фреймворка использована машина с предустановленной ОС семейства Linux. Такая конфигурация наиболее типична для разработки тестирования на языке Python, так как многие библиотеки изначально скомпилированы в данной среде. Самым популярным дистрибутивом ОС Linux в данный момент является Ubuntu. Данная версия в настоящее время наиболее распространена и поставляется в комплекте с GUI. Ее свойства приемлемы и достаточны для разработки системы автоматизированного тестирования.

В работе использован интерпретатор Python версии 2.7. Данная версия поддерживается большинством сторонних библиотек, что позволяет минимально ограничивать реализацию проекта (автоматизированного тестирования) в использовании сторонних компонентов.

На машине разработчика (или в среде выполнения тестов) возможно использование нескольких проектов на языке программирования Python. Однако, разными проектами могут использоваться разные библиотеки или одинаковые библиотеки, но разных версий. Так как библиотеки устанавливаются в систему глобально, может возникнуть конфликт версий программного обеспечения. Для решения данной проблемы хорошей мировой практикой считается использование изолированных окружений для каждого проекта. Для реализации этой концепции используем виртуальное окружение (virtualenv).

VirtualEnv находится в большинстве публичных репозиториев операционных систем. Установка данного пакета возможна с помощью стандартных пакетных менеджеров, входящих в состав поставки операционной системы.

При создании виртуального окружения происходит создание локальной версии интерпретатора, копирование всех его исходных файлов, а также глобально установленных в систему библиотек. Дальнейшая разработка фреймворка подразумевает установку необходимых библиотек локально в изолированную среду и, как следствие, отсутствие возможности использовать их глобально. При активизации окружения выполнение кода будет осуществляться непосредственно в данной среде.

Установка дополнительных пакетов в изолированное окружение осуществляется с помощью пакетных менеджеров, входящих в состав поставки Python: *easy_install* и *pip* . Названия и версии библиотек хранятся в специальном файле проекта *requirements.pip*. Организовывая зависимости данным образом и используя в дальнейшем менеджер пакетов *pip*, можно быстро развернуть окружение на любой другой машине. При копировании проекта в новом окружении необходимо запустить пакетный менеджер и передать ему в качестве параметра файл с требованиями – он автоматически

«скачает» из репозиториев необходимые библиотеки и установит их в текущее виртуальное окружение.

В нашем проекте реализации автоматизированного тестирования использованы два основных сторонних компонента: Selenium- для организации тестирования графических интерфейсов, а также Requests- для организации тестирования веб- ориентированных сервисов. Такой подход позволяет избежать конфликтов между версиями библиотек, упорядочить список используемых компонентов и обеспечить переносимость проекта на другие машины.

Согласно модели, описанной ранее, ядро фреймворка должно обеспечивать получение входных данных, собирать для выполнения тестовый набор, выполнять тесты и предоставлять отчет. Алгоритмы для реализации данных требований собраны в пакете CORE тестового фреймворка.

Схема пакета изображена на рисунке 1.

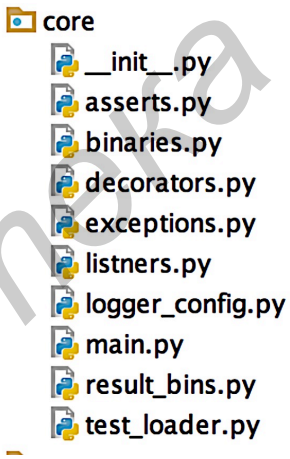

Рисунок 1 – Схема пакета тестового фреймворка

Тестовый фреймрок при запуске в режиме распределенного тестирования создает родительский процесс, осуществляет анализ тестового сценария и создает очередь с тестовыми сущностями. После чего создается пул процессов в рамках которых осуществляется выполнение тестовых сценариев. По мере окончания выполнения теста, процесс уничтожается, а отчет о тестовом прогоне поступает в родительский процесс, где агрегируется с результатами других процессов.

Для организации тестирования графических интерфейсов аннаользована библиотека Selenium Web Driver 2.0, портированная для языка программирования Python. Стандартные методы достаточно просты и обеспечивают лишь базовый функционал для работы с веб-браузером и веб-страницей. Так как одним из условий является создание многомодульной и

легкоподдерживаемой системы, то используем обертки над стандартной реализацией. Реализация, относящаяся к тестированию графического интерфейса, находится в пакете selenium_ui, который, в свою очередь, состоит из других.

Средства и логика для тестирования RESTсервиса собрана непосредственно в пакете ***rest***. Он обеспечивает работу с HTTP сервисом, используя библиотеку requests. Поскольку данная библиотека не входит в состав стандартной поставки, необходимо указать данную зависимость в файле требования requerements.pip. Структура данного пакета значительно проще, чем в случае с графическим интерфейсом, и состоит из двух модулей: client.py и resource.py.

Так как формат REST достаточно плохо стандартизирован в принципе, то подразумевается переопределение некоторых методов для работы с ресурсом, либо вообще создание другого класса. В любом случае, при тестировании REST, необходимо создавать максимально приближенный к описанному выше интерфейс для обеспечения гибкости и поддерживаемости используемого кода.

В соответствии с требованиями, предъявляемыми к программным продуктам, необходимо разработать логику для автоматизированных тестов. Для каждого тестового случая рекомендуется создать тестовый класс. Тестовые классы объединяют в группы (модули) в зависимости от тестируемого компонента. Далее все модули собираются в пакете tests.

Для запуска тестов необходимо наличие интерпретатора Python, развернутого виртуального окружения, установленных пакетов из файла требований, а также установленного в системе браузера MozzilaFireFox, так как он является браузером, используемым по умолчанию в библиотеке Selenium.

Поскольку язык программирования Python является интерпретируемым, то отсутствует необходимость его предварительной компиляции.

Тестовый набор собирается в файл tst_suite.xml. Запуск тестов осуществляется непосредственным выполнением в командной строке:

- pythonpyft.py –xml:<имя_файла> ,

где<имя_файла> - это tst_suite.xml для нашего случая.

Результат тестового прогона, а также отчет можно найти в файле log.log (в соответствии с конфигурацией системы логирования).

Для поддержания постоянного контроля над качеством разрабатываемого продукта в современном мире используются системы непрерывной интеграции. Хорошим примером такого сервиса является Jenkins - инструмент непрерывной интеграции, написанный на Java. Он

запускается в контейнере сервлетов, например таких, как ApacheTomcat или GlassFish. Jenkins поддерживает инструментарий для работы с разными системами контроля версий, включая CVS, Subversion, Mercurial, Git и Clearcase, может собирать проекты ApacheAnt и ApacheMaven, а также исполнять shell-скрипты и команды Windows.

После установки приложения на тестовом сервере необходимо произвести конфигурацию системы и определить триггеры для запуска тестовых сценариев. Запуск автоматизированных тестов может быть настроен на различные события, например, такие, как обновление исходного кода тестируемого продукта: после каждого изменения, внесенного разработчиком, будет проходить валидация качества. Также можно настроить запуск тестового сценария по расписанию, используя механизм, подобный cron: проводить тестирование программного продукта с определенной периодичностью. Выбор той или иной методики непрерывной интеграции зависит от подхода к разработке программного продукта, объему тестовых сценариев, их продолжительности, а также предъявляемых требований.

В результате практических исследований создан прототип тестового фреймворка, позволяющий проводить функциональное тестирование программного обеспечения по заранее созданным тестовым сценариям. За счет применения объектно-ориентированного подхода заложена архитектура, позволяющая в дальнейшем расширять функциональные возможности не изменяя уже существующий программный код. С помощью библиотек Seleniumi Request созданы прототипы для тестирования графических интерфейсов, а также веб- ориентированных сервисов. Данные подходы сочетают в себе как новизну в рамках использования языка программирования Python, так и лучшие мировые практики в создании фреймворков для автоматизированного тестирования.

Таким образом, в работе рассмотрены модели организации тестовых данных, тестовых наборов и тестовых случаев. Проанализированы и реализованы концепции алгоритмов для тестирования графических интерфейсов и удаленных веб- ориентированных сервисов. Показана возможность использования наиболее популярных шаблонов программирования в рамках языка Python.

В результате, разработанный итоговый фреймворк обладает простым и легко-читаемым кодом; легко расширяется и поддерживается; позволяет организовать тестирование графических интерфейсов и удаленных веб-ориентированных сервисов; при взаимодействии с системой непрерывной интеграции позволяет оперативно следить за качеством программного продукта. Разработанный фреймворк для автоматизированного тестирования

приложения позволяет проводить функциональное тестирование программных продуктов, эмулируя действия пользователя. С его помощью можно полностью проверить новую реализацию разработчиков на наличие ошибок в течение короткого промежутка времени, включая регрессионное тестирование созданных ранее функциональностей.

Результаты проведенных исследований и анализ полученного программного кода позволяют однозначно говорить о более высокой эффективности использования сверхвысокоуровневых языков программирования, таких как Python, в автоматическом тестировании с использованием модели распределенного выполнения тестов. Разработанный фреймворк перспективен с точки зрения внедрения и использования. Система достаточно легко поддается модификации и расширению. Альтернативой может быть использование материалов диссертации при составлении учебных программ и учебных планов в области тестирования и автоматизации тестирования.

По результатам проведенных исследований опубликовано 2 тезисов доклада:

1. Новиков В.И., Улащик А.Н. Автоматизация тестирования программного обеспечения средств связи с использованием SELENIUM 2 и системы распределенного выполнения тестов / Современные средства связи: Материалы Международн.научн.-практ. конф., Минск, 2013 г. – Минск: УО ВГКС, 2013. – с.157-158.

2. Новиков В.И., Улащик А.Н., Попова А.А. Автоматазиция тестирования программного обеспечения с использованием системы распределенного выполнения тестов // Управление информационными ресурсами: Материалы X Международн. науч.-практ. конф. – Минск: Академия управления при Президенте Республики Беларусь, 2013. – с.144-145.