АВТОМАТИЗИРОВАННОЕ ТЕСТИРОВАНИЕ КАК ПОДХОД К ОБЕСПЕЧЕНИЮ БЕЗОПАСНОСТИ, НАДЕЖНОСТИ И ЭФФЕКТИВНОСТИ ПРОГРАММНЫХ ПРОДУКТОВ

А.М. МОЖАЕВ

Пермский государственный национальный исследовательский университет, 614990, Пермь, Букирева, 15

Введение. Тестирования является одним из основных способов обеспечения качества программных продуктов. Тестирование программного продукта заключается в выполнении приложения на некотором множестве данных и сопоставлении полученных результатов с ожидаемыми (эталонными) с целью установить удовлетворяет ли приложение возложенным задачам[3].

Автоматизированное тестирование программного обеспечения – часть процесса тестирования на этапе контроля качества в процессе разработки программного обеспечения. [1]

Наиболее распространенной формой автоматизации является тестирование приложений через графический пользовательский интерфейс. Популярность такого вида тестирования объясняется двумя факторами: во-первых, приложение тестируется тем же способом, которым его будет использовать человек, во-вторых, можно тестировать приложение, не имея при этом доступа к исходному коду [3].

Существующие подходы к тестированию веб-приложений. Большинство подходов к тестированию являются тестированием по принципу «черного ящика». В распоряжении тестировщика имеется лишь тестируемое приложение и список требований, которым должно удовлетворять приложение. Задача тестировщика заключается в составлении сценария тестирования, которое обычно представляет собой переход по ссылкам внутри приложения, авторизация пользователей в системе, заполнение и расчет форм и документов, анализ получаемых страниц. Для автоматизации этих процессов существует множество инструментов позволяющих записывать и воспроизводить сценарий поведения пользователя в системе, однако ведущая роль при разработке тестов отводится человеку. Примерами таких инструментов являются:

1. Коммерческие: HP LoadRunner, HP QuickTest Professional, HP Quality Center, Segue SilkPerformer, IBM Rational FunctionalTester, IBM Rational PerformanceTester, IBM Rational TestStudio, AutomatedQA TestComplete
2. С открытым исходным кодом: Selenium, WATIR, Apache JMeter

Записанный единожды сценарий может далее многократно воспроизводиться автоматически. В случае внесения каких-либо изменений и доработок в приложение требуется проведение корректировки в скрипте сценария теста.

Тестирование прикладной многопользовательской системы. В работе представлено нагрузочное тестирование многопользовательской системы.

Тестируемая система реализована на основе клиент-серверной технологии с использованием СУБД MS SQL 2008 R2, интегрированной платформы и средств клиентского доступа, включающих в себя как «толстый» клиент (для функций централизованного администрирования, в том числе ведения реестра пользователей, разграничения прав доступа), так и «тонкий» клиент для основных функций Системы.

В качестве инструмента используемого для автоматизации нагрузочного тестирования использовался Apache JMeter v2.8, разрабатываемый Apache Software Foundation и распространяемый по лицензии Apahce Licens 2.0. [2]

Первым этапом при проведении нагрузочного тестирования формировался тест-план. Тест план или сценарий тестирования условно представляет собой описание некоторого бизнес-процесса, характерного предметной области, в которой функционирует система. Проще говоря, сценарий пользователя - это набор действий пользователя, выполняемых им в штатном режиме при работе с системой. Технически, набор действий пользователя в скрипте сценария представляет собой последовательность GET и POST HTTP-запросов. В JMeter сценарию пользователя соответствует объект "Test_plan".

Формирование сценария пользователя осуществляется двумя способами: в ручном режиме и в режиме записи.

В ручном режиме требуется "с нуля" создавать HTTP-запросы с их параметрами, создавать необходимые переменные, массивы, коллекции с данными, которые потребуются для запросов.

В режиме записи для формирования сценария используется специальный объект Record Controller (контроллер записи), который автоматически сохраняет последовательность HTTP-запросов, генерируемых при работе пользователя с приложением. В работе выбран второй способ, поскольку первый трудоемок и нецелесообразен.

Режим записи в JMeter реализован через использование встроенного в JMeter прокси-сервера. Для осуществления записи требуется добавить объект «HTTP Proxy Server», указать для него порт и режим группировки

запросов при записи, а также фильтрацию запросов по их типу. Кроме этого требуется добавить контролер записи «Record Controller», в который будет собирать в себе объединенные в транзакционные группы HTTP-запросы. После добавления компонент ,требуется перенастроить браузер для работы с данным прокси серверов, указав в настройках браузера порт данного прокси сервера. Кнопкой «Start» запускаем процесс записи сценария и начинаем выполнять пользовательские действия с системой. По завершению записи необходимо остановить работу прокси сервера.

Вторым шагом осуществляется корректировка скрипта сценария тестирования. В процессе записи в элементе «Record Controller» сгенерировалось множество запросов, содержащих в себе метаданные, в виде HTTP заголовков (HTTP Header). Среди множества запросов могли записаться «лишние» запросы, URL которых отличен от URL-адреса приложения. Данные элементы требуется удалить из сценария.

Любое веб-приложения многопользовательское. Чтобы каждый пользователь смог корректно войти в систему и работать в ней в процессе воспроизведения сценария, требуется обеспечить уникальность сессии пользователя. В JMeter для этого предусмотрены компоненты групп Post- и Pre- Processor: массивы, коллекции, динамические переменные, регулярные выражения. При входе первого пользователя в приложение осуществляется обработка ответа сервера на запрос и извлекается из него с помощью регулярного выражения идентификатор сессии пользователя, который далее будет передаваться в качестве параметра запроса следующим пользователем при входе в систему.

Вся действия пользователей и логика их работы объединены в компоненте «Tread Group». В этом же компоненте задаются параметры нагрузки на приложение: количество пользователей, интервал времени входа между двумя пользователями, задержки между входами и количество повторений. Так же предусмотрено множество других элементов, позволяющих моделировать реальную работу пользователей: циклы, условные операторы, таймеры, генераторы случайных задержек.

Для мониторинга и анализа результатов предусмотрены компоненты группы Listener, позволяющие вести достаточно подробную аналитику работы системы. Данные компоненты позволяют вести и сохраять статистику загрузки сервера, вести логирование ошибок приложения, строить графики зависимости различных показателей от числа пользователей системы, на временном интервале, контролировать выполнение функциональных частей приложения и их среднее время отклика, задержку и дру-

гие статистические показатели.

Результаты. Применение автоматизированного нагрузочных тестов позволило определить оптимальную конфигурацию веб сервера и сервера баз данных, на которых планируется развернуть веб-приложения, для заданного количества пользователей. Удалось обнаружить ошибки в работе отдельных функциональных частей приложения в процессе прогонки нагрузочных тестов в многопользовательском режиме. Удалось локализовать наиболее медлительные компоненты и части системы, установить зависимость среднего времени отклика компонент системы, нагрузки серверов и используемой памяти от числа авторизованных пользователей.

Заключение. Средства автоматизированного тестирования позволяют решать ряд вопросов:

- Экономия ресурсов и повышение качества тестирования. Автоматическое тестирование по заданному сценарию не требует участия человека – система сама тестирует программный продукт во всех нужных режимах. Вмешательство человека требуется только для пополнения библиотеки сценариев, изучения отчетов и поддержки тестов в актуальном состоянии.
- Стабилизация надежности. Когда в системе проводятся какие-либо изменения, то самые трудноуловимые ошибки – те, что возникают в уже проверенных компонентах. Повторный запуск сценариев тестирования после внесения изменений позволяет обнаружить ошибки в ситуациях, где тестировщики с большой вероятностью мог бы их пропустить.

СПИСОК ЛИТЕРАТУРЫ

1. Автоматизированное тестирование программного обеспечения // Википедия – свободная энциклопедия.
   URL: http://ru.wikipedia.org/wiki/Автоматизированное_тестирование
2. JMeter // Википедия – свободная энциклопедия.
   URL: http://ru.wikipedia.org/wiki/JMeter
3. *Силаков Д. В.* АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ WEB-ПРИЛОЖЕНИЙ, ОСНОВАННЫХ НА СКРИПТОВЫХ ЯЗЫКАХ // Труды ИСП РАН. 2008. №2.
   URL: http://cyberleninka.ru/article/n/avtomatizatsiya-testirovaniya-web-prilozheniy-osnovannyh-na-skriptovyh-yazykah