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

УДК 004.054

Дашукевич
Владимир Павлович

Автоматизация функционального тестирование Web-приложений

АВТОРЕФЕРАТ
на соискание академической степени
магистра технических наук

по специальности 1-40 80 05 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Научный руководитель
Скудняков Ю.А.
к.т.н., доцент

Минск 2014

КРАТКОЕ ВВЕДЕНИЕ

Возрастающая важность Web-проектов для организаций за последние годы стала очевидной. Быстрое и успешное внедрение таких систем часто является критически важным для бизнес-стратегии многих организаций – в основном в части того, как они взаимодействуют с заказчиками, клиентами или бизнес-партнерами.

Единственным средством минимизации этих рисков, связанные с неизбежно возникающими при создании систем такой сложности ошибками, является проведение на всех этапах создания или внесения изменений в ПО тестирования на соответствия требованиям и выполнению базовой функциональности. В общем случае тестирование представляет собой наблюдение за работой системы и анализ ее правильности в ряде специально создаваемых ситуаций с учетом всех существенных аспектов ее поведения. Также стоит отметить, что, с точки зрения затрат на выпуск программного обеспечения, исправление ошибки сделанной при тестировании системы в 10-100 меньше, чем после выпуска, разрабатываемого ПО.

Автоматизация тестирования снижает время повторного тестирования, необходимое на проверку регрессий в уже написанном коде, и избавляет от человеческого фактора, но, в свою очередь, во много раз увеличивает затраты на разработку и поддержку написанных тестовых сценариев.

Таким образом, рассмотрение задачи по автоматизации тестирования, проверке всей базовой функциональности и генерации тестовых сценариев для программных интерфейсов Web-приложений без участия тестировщиков и программистов является весьма актуальной задачей.

Цель настоящей работы заключается в разработке метода автоматизации функционального тестирования Web-приложений, генерации последовательности тестовых сценариев для организации регрессионного тестирования, а также формированию полного отчетов о полученных в процессе тестирования ошибках и исключениях.

В работе рассматривается разработка программного обеспечения для функционального тестирования программных интерфейсов Web-приложений, которое самостоятельно определяет элементы управления и взаимодействует с ними, моделируя поведение конечного пользователя, тем самым уменьшая затраты и время на разработку и тестирования программного обеспечения. Учитывая достоинства и недостатки уже существующих решений, в данной работе предлагается метод для организации автоматического тестирования на основе конечных автоматов, а также разработка алгоритмов генерации тестовых сценариев с применением теории графов. Также были применены принципы объектно-ориентированного программирования и элементы алгоритмической теории при разработке программной модели алгоритма генерации.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Цель и задачи исследования

Целью диссертационной работы является разработка метода и алгоритма для автоматизации функционального тестирования Web-приложений, генерации последовательности тестовых сценариев для организации регрессионного тестирования, а также формированию полного отчетов о полученных в процессе тестирования ошибках и исключениях.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести анализ известных подходов к автоматизации тестирования интерфейса программирования web-приложения.
2. Определить возможность полной автоматизации функционального тестирования Web-приложения.
3. Разработать метод и алгоритм для обхода все страниц Web-приложения и взаимодействия со всеми активными элементами интерфейса.
4. Разработать алгоритм обхода модели для генерации тестов.
5. Реализовать совокупность программных средств, объединенных в универсальную среду тестирования программных интерфейсов web-приложений.
6. Реализовать возможность создания автоматических отчетов об ошибках в уже существующих системах отслеживания ошибок.
7. Провести экспериментальные исследования и апробацию разработанного метода.

Объектом исследования являются методы и средства автоматизации процесса функционального тестирования Web-приложений.

Предметом исследования являются программное обеспечение для решения задач автоматизации функционального тестирования Web-приложений, методы и алгоритмы обхода всех страниц приложения и взаимодействие со всеми функциональными элементами приложения.

Основной *гипотезой*, положенной в основу диссертационной работы, является возможность автоматизации функционального тестирования Web-приложений, генерации последовательности тестовых сценариев для организации регрессионного тестирования, а также формированию полного отчетов о полученных в процессе тестирования ошибках и исключениях без участия тестировщика и программиста. Все Web-приложения состоят из страниц, связанных между собой ссылками и имеющими некоторые функциональные элементы для пользовательского воздействия. В общем случае для функционального тестирования такого рода ПО можно использовать программу обходчик, которая моделирует взаимодействие пользователя со всеми активными элементами на странице, вводит различные данные в текстовые поля для проверки, анализирует полученные результаты, записывает все свои действия для генерации регрессионных тестов, анализирует временные задержки между запросами на сервер и ответами, а также генерирует

автоматические отчеты о полученных ошибках и не пройденных тестах. Также с помощью данного ПО возможна организация нагрузочного и стрессового тестирования, когда моделируется группа пользователей использующая Web-сервис.

Личный вклад соискателя

Результаты, приведенные в диссертации, получены соискателем лично. Вклад научного руководителя Ю.А. Скуднякова, заключается в формулировке целей и задач исследования.

Апробация результатов диссертации

Основные положения диссертационной работы докладывались и обсуждались на XIX международной научно-технической конференции «Современные средства связи» (Минск, Беларусь, 2014); XVIII международной научно-технической конференции «Современные средства связи» (Минск, Беларусь, 2013).

Опубликованность результатов диссертации

По теме диссертации опубликовано 2 работы в сборниках трудов и материалов международных конференций.

Структура и объем диссертации

Диссертация состоит из введения, общей характеристики работы, четырех глав, заключения, списка использованных источников, списка публикаций автора и приложений. В первой главе представлен анализ уже существующих подходов, методов и программных решения для автоматизации функционального тестирования Web-приложений. Вторая глава посвящена разработке архитектуры ПО и алгоритмов для проведения автоматизации тестирования, обходе всех существующих страниц приложения, оценке значимости активных элементов и генерации регрессионных тестов. В третьей главе предложены методы автоматизации тестирования, нахождения функциональных элементов, определения классов эквивалентности для входных данных, а также практическая реализация данного ПО. В четвертой главе представлены результаты экспериментальных исследований для различного вида Web-приложений, а также приводятся рекомендации по практическому использованию результатов.

Общий объем работы составляет 78 страниц, из которых основного текста – 65 страниц, 9 рисунков на 9 страницах, 2 таблицы на 2 страницах, список использованных источников из 46 наименований на 4 страницах и 5 приложений на 9 страницах.

ОСНОВНОЕ СОДЕРЖАНИЕ

Во **введении** определена область и указаны основные направления исследования, показана актуальность темы диссертационной работы, дана краткая характеристика исследуемых вопросов, обозначена практическая ценность работы.

В **первой главе** проведен анализ уже существующих решений, используемых при автоматизации функционального тестирования Web-приложений. Выполнен анализ применяемых методов и алгоритмов автоматизации функционального тестирования Web-приложений.

Были рассмотрены следующие способы автоматизации тестирования и программное обеспечение для них:

1. Запись и воспроизведение. В данном случае используются инструменты для записи действий тестировщика во время ручного тестирования
2. Написание сценария. В данном случае поведение пользователя описывается с помощью определенных команд.
3. Тестирование управляемое данными. Особенностью является то, что тестовые скрипты выполняются и верифицируются на основе данных, которые хранятся в центральном хранилище данных или базе данных.
4. Тестирование по ключевым словам. В этом случае конечный тест представляет собой не программный код, а описание последовательности действий с их параметрами.

Функциональное тестирования Web-приложения также может осуществляться на уровне его взаимодействия с сервером. Для автоматизации данного метода также может быть использованы написанные сценарии. В данном случае сценарий описывает не пошаговое взаимодействие пользователя с интерфейсом приложения, а набор HTTP запросов, тем самым создает иллюзию присутствия клиентской части приложения. Вместе с этим можно использовать технологию UniTesK для определения pre и post действий и принятии решений о правильность работы приложения.

Вторая глава посвящена методу автоматизации тестирования пользовательского интерфейса Web-приложений, который основан на применении расширенных конечных автоматов. Суть метода заключается в выделении состояний конечного автомата путем декомпозиции приложения и последующей генерации тестов с применением алгоритма траверса графов.

В контексте функционального тестирования Web-приложений на системном уровне, взаимодействие конечного пользователя с приложением предложено разбить на следующие типы.

Взаимодействие с элементом пользовательского интерфейса, который не предусматривает ввод данных (детерминированный элемент). Множество различных детерминированных элементов напрямую зависят от разнообразия типов элементов интерфейса и способов действия на них и является конечным числом.

Взаимодействие с элементом пользовательского интерфейса, который предусматривает ввод данных (вариационный элемент). В общем случае пользователь может ввести в приложение любой набор данных, это предполагает бесконечное число подобных действий ввода данных.

Определив конечный набор детерминированных действий, и ограничив бесконечный набор вариационных действий конечным набором, можно свести взаимодействие пользователя с web-приложением к детерминированному, конечному набору операций.

После выбора определенного элемента, нам необходимо выбрать определённое воздействие на него. Так как каждый активный элемент Web-страницы имеет несколько возможных способов взаимодействия конечного пользователя с ним, то нам следует также оценить важность каждого из них. В данном случае мы будем использовать упрощенную формулу определения вероятности того или иного воздействия:

$$P_a = \frac{P_a^{\text{исп}}}{\sum_b P_b^{\text{исп}}}, \quad (1)$$

где P_n - вероятность выполнения n-ого воздействия над найденным элементом в тесте,

$P_n^{\text{исп}}$ - вероятность использования конечным пользователем n-ого воздействия над найденным элементом Web-страницы.

Также приведен расчет вероятности появления ошибки на одном из возможных сценариев использования тестируемого Web-приложения пользователем.

$$Q = 1 - P = 1 - \sum_{i=1}^{M} \pi_i * \prod_{j=1}^{K} (1 - q_{ij}), \quad (2)$$

В общем случае при проверке всех возможных маршрутов для Web-приложения вероятность появления ошибки на одном из возможных сценариев использования будет равняться 0. Но при этом сложность и количество тестов будет слишком большим и, следовательно, время на прогон всех этих тестов также будет недопустимо большим.

С учетом весов функциональных элементов (при этом все виды воздействий считаются одинаково важными) формула по оценке эффективности тестирования с помощью разрабатываемого программного обеспечения будет выглядеть следующим образом:

$$L = \frac{\sum_{i=1}^{M} a_i * P_i}{\sum_{i=1}^{N} a_i * P_i}, \quad (3)$$

где P_i – вероятность использования данного функционального элемента конечным пользователем,

M – количество найденных функциональных элементов,

N – общее количество функциональных элементов Web-страницы или Web-приложения,

a_i – количество возможных способов взаимодействия с i-ым функциональным элементом, переводящих ПО в другое состояние.

Данная формула позволяет нам оценить процент протестированной базовой функциональности Web-приложения.

В **третьей главе** рассмотрена практическая реализация ПО для автоматизации тестирования пользовательского интерфейса Web-приложений и генерации регрессионных тестов.

Для реализации данного ПО весь процесс тестирования был разбит на три составные части: обход приложения и выделения основных web-страниц, обход заданной страницы и поиск всех активных элементов и выделения всех возможных способов взаимодействия с активными элементами интерфейса.

Для решения первой задачи нам необходимо обойти все возможные страницы, при чем стоит различать страницы с различным контентом, но по факту представляющие динамически генерируемую с помощью шаблонов страницу, так как для некой информации из БД могут генерироваться совершенно разный контент и таких страниц может быть очень много. Для решения данной проблемы существует методика снимка HTML.

Снимок HTML – это все содержание, отображаемое на странице после выполнения кода JavaScript. Для этого необходимо чтобы Web-приложение поддерживало схему сканирования AJAX. В данном случае сам сервер будет выдавать снимок HTML для каждого URL AJAX, который виден пользователю. После этого программа для обхода страниц может найти в этом снимке ссылки на новые страницы и пополнить таблицу тестируемых страниц Web-приложения.

Для определения схожести новой страницы с уже включенной в список тестируемых страниц можно использовать её URL и HTML код. Для начала высчитаем два типа степеней схожести между новой страницей и страницами из списка страниц для тестирования:

1. Схожесть URL страниц. Если длина двух URL одинаковая или отличается не более чем на 2 символа (такое условие необходимо для идентификаторов сущностей БД), то схожесть между URL прямопропорциональна количеству совпавших символов. Если длина разная, то расстояние пропорционально разности длин.

2. Схожесть DOM дерева страницы. Расстояние между двумя DOM деревьями — разность количества одноименных тегов. Так как дерево может динамически достраиваться, то программа обходчик должна дождаться окончания всех AJAX запросов на данной странице.

Для решения второй задачи по обходу заданной страницы и поиску всех элементов в качестве модели сложной формы с набором активных элементов был взят «граф состояний». Каждая вершина графа — это состояние формы. Оно может измениться за счет удаления поля ввода из формы, изменения значения поля ввода или добавления нового.

На основании выдвинутых предположений была разработана последовательность действий для тестирования web-страниц с набором активных элементов, объединённых в форму:

1. В самом начале нам необходимо определить все активные элементы формы. Пусть изначально форма состоит из элементов $A_1...A_n$.
2. Далее, начиная с самого верхнего поля (имеется ввиду физическое расположение полей на странице) пробуем повзаимодействовать с A_i всеми возможными способами. Пусть для каждого активного элемента имеется k способов взаимодействия.
3. Зафиксируем те элементы $B_1...B_k$, которые появлялись при различных заполнениях элемента A_i.
4. Рекурсивно повторить процедуру для каждого из B_j.

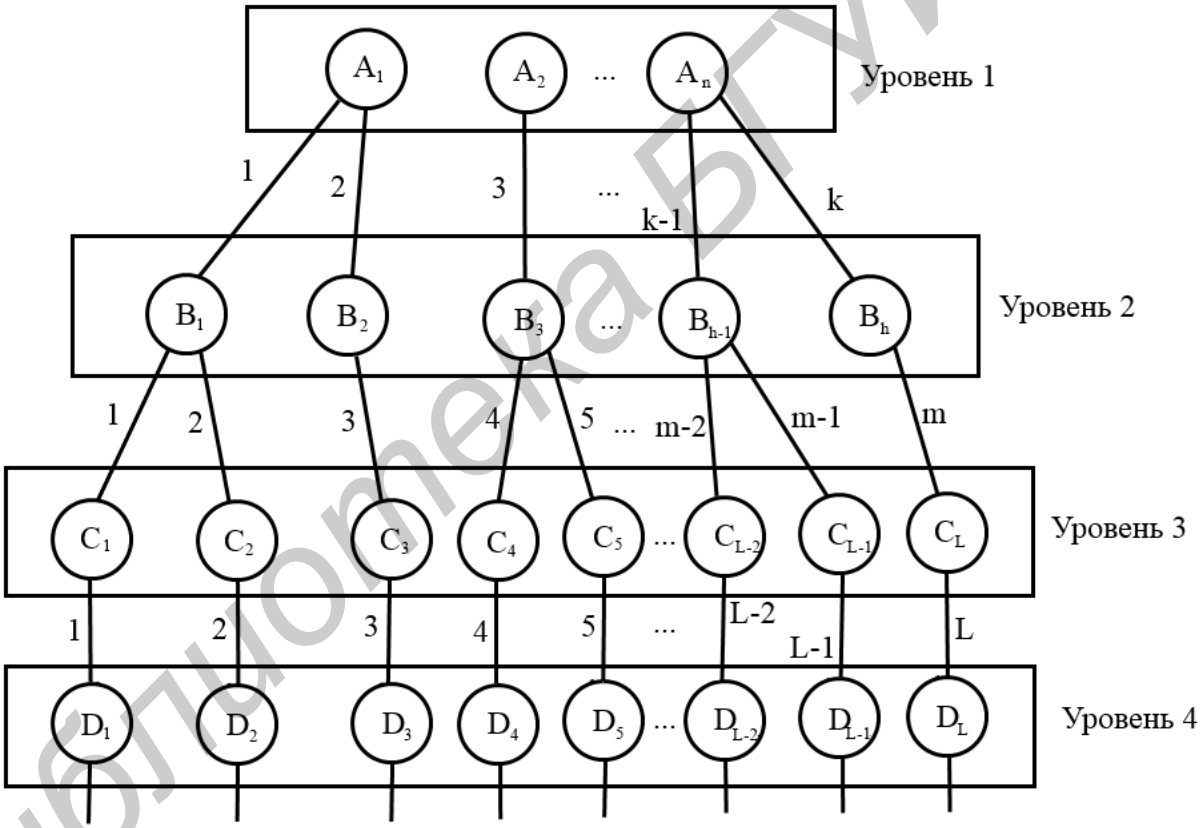

Рисунок 1 – Алгоритм тестирования активных элементов Web-страницы

При таком подходе алгоритм может работать очень долго, поскольку количество способов взаимодействия будет расти экспоненциально от числа элементов. Для того чтобы программа для тестирования форм могла закончить процесс тестирования за разумное время, перебор всех возможных способов взаимодействия осуществляется только для элементов, находящихся на глубине не более 2, а для всех остальных выбираем одно случайное значение. Такой подход необходим для тестирования форм, у которых необходимо заполнение некоторого количества обязательных полей.

В следствии данного допущения, после взаимодействия с первым набором активных элементов $A_1...A_n$ k способами, получив $B_1...B_h$ новых элементов, мы также взаимодействуем с ними всеми возможными m способами. После этого с каждым элементом, который становится доступен при заполнении B_i, взаимодействуем единственным случайным образом.

Оптимальное покрытие генерируется из расчета, что A_i — выпадающий список, у которого вариантов столько, сколько из него путей до уровня 3. Конкретный пример графа состояний приведен на рисунке 1.

Для увеличения эффективности функционального тестирования для каждого вариационного элемента web-страницы необходимо выбрать соответствующие ему классы эквивалентности вводимых данных. В общем случае данная задача сводится к максимизации распознавания полей ввода данных. Самым простым решением данной проблемы является составление словаря типов. То есть найдя текстовое поле, программа тестировщик должна проверить окружающие элементы на предмет меток и надписей для пользователя. Найдя их, она должна обратиться к словарю, при наличии подобного элемента, получить классы эквивалентности для вводимых данных и способы взаимодействия. При отсутствии в словаре какой-либо информации о такого рода элементах, принять его за обычное текстовое поле.

В качестве ключевых слов для всех типов из словаря используются те, которые, присутствуя рядом с данным полем в интерфейсе, повышают вероятность того, что оно относится к данному типу. Разумеется, не все ключевые слова одинаково полезны. Каждому ключевому слову сопоставлен вес.

Данный алгоритм позволяет за разумное время обойти все функциональные элементы Web-приложения, протестировав их на различных наборах данных (как валидных, так и ошибочных).

В **четвертой главе** проведен анализ полученных результатов по автоматизации функционального тестрования трех типов Web-приложений. Были проведены эксперименты на реально работающих проектах.

Для начала использовался стандартный Web-сайт магазина, сделанного на основе таких конструкторов сайтов как wix, google.sites и т.д. Покрытие базовой функциональности элементов управления интерфейсом была равна практически 100%. При этом сгенерированные последовательности действий для регрессионного тестирования были довольно короткими, следовательно, и сложность самих тестов не большая.

Вторым приложением для тестирования был также Web-сайт магазина, но в данном случае использовался шаблон для CMS Drupal. Многие стандартные модули любых общеизвестных CMS имеют высокую эффективность тестирования с помощью разработанного программного обеспечения, так как имеют хорошую архитектуру интерфейса, стандартные элементы управления, подписи для каждого поля ввода и простую логику работы. Результаты тестирования были такими же, как и в первом случае, так как тестируемый Web-сайт был построен из стандартных модулей.

Третьим приложением для тестирования было одностраничное Web-приложение, большая часть бизнес логики которого перенесена на клиентскую сторону. Web-сервер использовался для получения данных и шаблонов страниц. Для него использовалась REST архитектура, то есть все данные загружались по вызову удаленной процедуры, которая представляет собой обычный HTTP запрос, а необходимые данные передавались в качестве параметров запроса. На стороне клиента использовались библиотеки и фреймворки для построения интерфейсов: AngularJS, JQuery и JQueryUI.

Весь базовый функционал основных форм и активных элементов был протестирован за допустимое время (около 2 часов). Все рутинные действия, такие как введение в числовое поле не числовых символов, текста для XSS атаки и так далее также были протестированы и покрыты регрессионными тестами, что существенно сэкономило затраты на первичное и повторное ручное тестирование данной функциональности.

Также в процессе тестирования и применение данного программного обеспечения на реальных Web-проектах проявились незапланированные способы использования разработанного программного обеспечения. В частности, с помощью данного ПО можно проводить нагрузочное тестирование клиентской и серверной части Web-проекта для выявления ошибок связанных с состоянием гонки, то есть ошибка проектирования системы или приложения, при которой работа системы или приложения зависит от того, в каком порядке выполняются части кода, и времени отклика приложения на запросы, отправленные из браузера.

При таком способе тестирования исправления ошибок приложения увеличивается вероятность обнаружения других неисправностей, не только связанных со свойствами отдельных функций, но и с общей функциональностью системы, предоставляемой через интерфейс. В общем случае использование конечно-автоматной модели обеспечивает прозрачность структуры и поведения разрабатываемых тестовых сценариев.

ЗАКЛЮЧЕНИЕ

В начале данной работы была поставлена задача разработки программного обеспечения для автоматизации функционального тестирования программных интерфейсов Web-приложений, которое самостоятельно определяет элементы управления и взаимодействует с ними, моделируя поведение конечного пользователя, тем самым способствует уменьшению затрат и времени на разработку и тестирования приложений.

В настоящей работе на основе литературных источников и уже существующих программных решений изложено современное состояние и тенденции развития методов и средств для автоматизации функционального тестирования. Выделено четыре основных направления: запись действий тестировщиков, написание сценариев, взаимодействующих с интерфейсом Web-

приложения, тестирование управляемое данными и тестирование по ключевым словам.

Учитывая достоинства и недостатки уже существующих решений, в данной работе предлагается метод для организации автоматического тестирования на основе конечных автоматов, а также разработка алгоритмов генерации тестовых сценариев с применением теории графов. Также были применены принципы объектно-ориентированного программирования и элементы алгоритмической теории при разработке программной модели алгоритма генерации.

В качестве метода автоматизации функционального тестирования Web-приложений предложен метод, основанный на применении расширенных конечных автоматов. Суть метода заключается в выделении состояний конечного автомата путем декомпозиции приложения и последующей генерации тестов с применением алгоритма траверса графов. На его основе был разработан алгоритм, который без участия тестировщика обходит все страницы Web-приложения и взаимодействует со всеми функциональными элементами. Преимуществами данного алгоритма по сравнению с приведёнными способами автоматизации тестирования Web-приложений являются:

1. Автоматический обход приложения, поиск активных элементов, определение важности и классов эквивалентности для входных данных для каждого из них и взаимодействия с ними. В данном случае нет необходимости в тестеровщике для проведения тестирования базовых элементов Web-страниц;
2. Автоматическая генерация тестовых сценариев для проведения регрессионного тестирования. При этом для каждой новой версии Web-приложения можно перегенерировать весь набор регрессионных тестов заново.

На основании проведенных исследований было разработано программное обеспечение выполняющее следующие функции:

1. Автоматическое взаимодействие со всеми элементами интерфейса Web-приложения;
2. Тестирования активных элементов различными входными данными;
3. Тестирования базовой функциональности всех стандартных элементов HTML страниц;
4. Генерация регрессионных тестов на основании проделанных действий и различных входных данных для дальнейшего усовершенствования тестировщиками;
5. Мониторинг потребления памяти, AJAX запросов и ответов, и процессорного времени;
6. Генерация отчета об обнаруженных в процессе тестирования ошибках с подробным описанием последовательности действий.

Кроме основных возможностей по тестированию разработанное программное обеспечение может выполнять дополнительные функции:

1. Для тестирования мобильных версий Web-приложений;
2. Нагрузочное и стрессовое тестирование.

Для проведения экспериментов были выбраны несколько Web-приложений различной сложности. Все они были типичными представителями

существующих в сети интернет приложений, так как для их создания использовались общедоступные конструкторы сайтов, CMS, фреймворки и библиотеки. Результаты доказали эффективность использования данного программного обеспечения, даже если тестируемое приложение состоит не из стандартных модулей и компонентов.

СПИСОК ОПУБЛИКОВАННЫХ РАБОТ

1-А. Дашукевич, В.П. Автоматизация функционального тестирования web-приложений / В.П. Дашукевич // Материалы XVIII Междунар. науч.-техн. конф. "Современные средства связи". – Минск: ВГКС, 2013. – с. 170.

2-А. Дашукевич, В.П. Автоматизация функционального тестирования web-приложений / В.П. Дашукевич // Материалы XIX Междунар. науч.-техн. конф. "Современные средства связи". – Минск: ВГКС, 2014. – с. 156.