$$P = \begin{pmatrix} 0.1 & 0.9 \\ 0.9 & 0.1 \end{pmatrix}, \ \pi = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}.$$

Моделировались цепи Маркова, затем методом Монте-Карло оценивался риск прогнозирования их будущих значений при горизонте прогнозирования $\tau = 1$, вычислялись оценки $r^+(T) = r_0 + A/T$ (где константа A определена в правой части (3)). Число прогонов в методе Монте-Карло $K=10^5$. Результаты экспериментов приведены на рисунке 1.

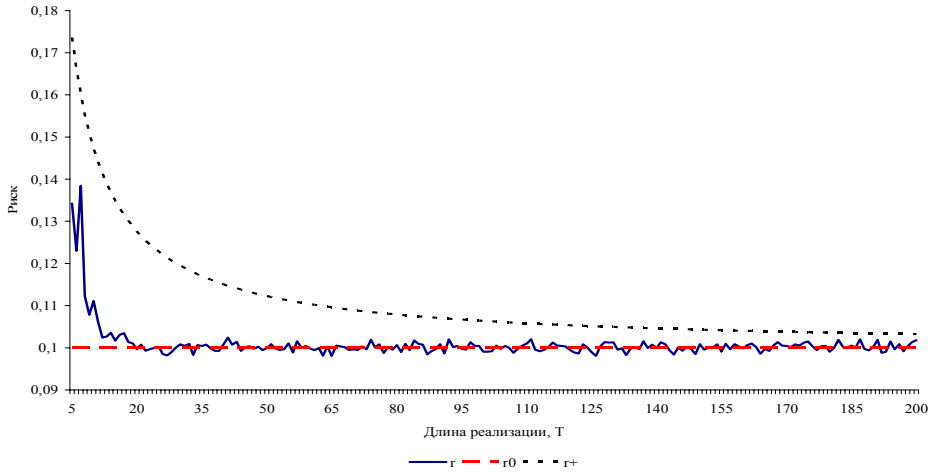

Рис.1. Графики оценок риска

В ходе численных экспериментов получено, что точность главного члена оценки (3) существенно зависит от матрицы P. Эта точность ниже, если матрица вероятностей переходов P имеет доминирующие диагональные элементы, по сравнению со случаем, когда диагональные элементы не являются доминирующими.

Литература

1. *Basawa I. V., Rao B. L. C.* Statistical interference for stochastic processes. N.Y. // Academic Press. 1980.
2. *Дуб Дж*. Вероятностные процессы. // М. ИЛ. 1956.
3. *Харин Ю. С.* Оптимальность и робастность в статистическом прогнозировании. // Мн. БГУ. 2008.
4. *Харин Ю. С.* Вероятностно-статистический анализ цепей Маркова высокого порядка // Вестн. Белорус. ун-та. Сер. 1. 2006. N 3. С. 80–86.

ТЕСТИРОВАНИЕ WEB-ПРИЛОЖЕНИЙ

А. Г. Морейнис

В настоящее время все большее распространение получают приложения, взаимодействующие с пользователем посредством Web-интерфейса.

Поэтому актуальным вопросом для Web-приложений является обеспечение качества работы, т.е. программы должны выдавать те страницы, которые ожидает увидеть пользователь. Таким образом, необходимо решать задачу функционального тестирования, например, с помощью нагрузочного тестирования. Большинство Web-приложений постоянно развиваются и модифицируются, поэтому важным является наличие регрессионных тестов для того, чтобы удостовериться, что результаты внесения изменений не нарушают функциональности приложения. Непосредственное тестирование Web-приложений человеком отнимает много времени и в случае больших приложений малоэффективно, т.е. переход по различным ссылкам внутри приложения и анализа отображаемых страниц, работа с большим количеством входных параметров, работающих с базами данных сложной структуры и с большим количеством записей. При разработке таких программных продуктов вопрос об автоматизации процесса их тестирования стоит особенно остро.

Одними из самых распространенных видов тестирования Web-приложений являются [4, 5]:

- Тестирование Web-приложений «на проникновение» основано на проверке ввода на корректность.
- Функциональное тестирование или тестирование черного ящика– это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, т.е. способности программного обеспечения в определенных условиях решать задачи, которые нужны пользователям.
- Использование Selenium для тестирования Web-приложений на различных браузерах представляет собой инструментальное средство приемочного тестирования, тесты которого выполняются непосредственно в браузере.
- Тестирование Web-приложений с помощью Ruby – это высокоуровневая методика тестирования Web-приложений, которая используется как для системного, так и для приемочного тестирования. Ключевую роль при этом играет Ruby-библиотека Application Testing In Ruby, которая позволяет запрограммировать действия браузера Internet Explorer на языке Ruby так, чтобы можно было автоматизировать значительную часть ручной работы тестеров по заполнению форм, переходу по ссылкам и т.д.
- Нагрузочное тестирование Web-приложений – это запись трафика, который генерируется при общении локального компьютера с сервером, а затем использует записанный трафик для эмуляции действий пользователей.

- Приемочное тестирование основано на тестировании группы связанных классов, которые создают желаемый результат;
- Применение инструментов браузеров Internet Explorer, Mozila Firefox, Opera, Google Chrome ориентированы на Web-разработчиков для создания, редактирования и анализа Web-страниц.
- Проанализировав различные виды тестирования Web-приложений, мною было разработано тестирующее Web-приложение, которое предназначено для Web-разработчиков с целью оптимизации рабочего процесса. Разработанное Web-приложение основано на следующем:
- Структуру любого Web-приложения можно представить в виде дерева или любой другой древовидной структуры;
- Для анализа функциональности Web-приложения потребуется следующее [3]:
    - кнопки;
    - гипертексты;
    - картинки;
    - JavaScript;
    - URL;
    - links;
    - формы.
- Информация страницы содержит следующее:
    - все возможные пути переходов с текущей страницы;
    - информация о странице;
    - структурное содержание;
    - время обработки;
    - пройденные и не пройденные тесты.
- Настройки:
    - ограниченное или неограниченное количество тестов;
    - точка входа;
    - предмет тестирования;
    - корректные и некорректные параметры;
    - установка маршрута для тестирования конкретной области web-приложения;
    - тестирование определенных страниц;
    - задание количества пользователей с личными настройками.

Выше изложенный подход тесно пересекается с тестированием Web-приложений «на проникновение», с помощью Ruby, с использованием Selenium, с помощью инструментов браузеров Internet Explorer, Mozila Firefox, Opera, Google Chrome посредством ссылок, полей для ввода, гипертекстов, форм.

Для реализации проекта я использовал Ext GWT [1, 2], что позволило ускорить разработку проекта, уменьшило затраты на разработку интерфейса. Проект разделен на две части: серверную и клиентскую, взаимодействие между которыми происходит в асинхронном режиме, что позволяет улучшить передачу данных и показать работоспособность программы разработчику. Для сохранения и получения важной информации используется MySQL, в котором хранится конфиденциальная информация пользователей сайтов. Разработанное Web-приложение позволяет программисту ускорить разработку web-приложения, т.к. помогает найти ошибки на ранних этапах разработки, что увеличивает устойчивость и уменьшает стоимость проекта. Также Web-приложение позволяет получить абстрактное представление о структуре разрабатываемого или тестируемого web-приложения, которое даёт лучшее представление и отделяет разработку от тестирования. Так разработчику не придётся тратить время на ожидание работы сотрудника по тестированию Web-приложения и избавляет самого от тестирования, что позволяет сконцентрироваться на разработке, устранении неполадок. В результате автоматизации процесса тестирования Web-приложений устраняется человеческий фактор в поиске существующих ошибок, что позволяет сократить время разработки и поддержки самого проекта и отпадает необходимость привлечения дополнительных ресурсов.

При тестировании Web-приложений необходимо грамотно выбирать инструменты для их тестирования, поскольку этот выбор сделает более эффективной работу, как разработчика, так и сотрудника по тестированию Web-приложений.

С моей точки зрения, тестирование web-приложений “на проникновение”, с помощью Ruby, с использованием Selenium, с помощью инструментов браузеров IE, Mozilla Firefox, Opera, Google Chrome лучше всего подходят для тестирования Web-приложения на клиентской стороне на начальном этапе, что позволяет без перегрузки сервера сократить время на исправление, добавление информации в Web-приложении. А функциональное, нагрузочное, приёмочное виды тестирования больше всего подходят для серверной части Web-приложения и должны применяться на первых этапах разработки, т.к. правильная разработка на данных этапах не приводит к увеличению стоимости проекта и появлению более серьёзных ошибок.

Разработанное Web-приложение тесно связано с тестированием Web-приложений «на проникновение», с помощью Ruby, с использованием Selenium, с помощью инструментов браузеров IE, Mozilla Firefox, Opera, Google Chrome.

Литература

1. *Grant K. S*. Developing with Ext GWT Enterprise RIA Development. Apress. 2009.
2. *Dewsbury R*. Google Web Toolkit. Prentice Hall. 2007.
3. *Фридл Дж*. Регулярные выражения. Символ-Плюс. 2008.
4. *Брайан А*. Тестирование и оптимизация веб-сайтов: руководство по Google Website Optimizer. Диалектика. 2009.
5. *Стотлемайер Д.* Тестирование Web-приложений. КУДИЦ-Образ. 2003.

ЗАДАЧА ОПТИМАЛЬНОГО ПОКРЫТИЯ ИЗМЕНЕННЫХ ДАННЫХ

Столяров В. О., Шавлак М. Ю.

ВВЕДЕНИЕ

Задача оптимального покрытия измененных данных рассматривается при передаче модифицированного изображения с экрана одного устройства на экран другого устройства и является частным случаем задачи покрытия множества. Для ее решения были рассмотрены следующие алгоритмы:

- Сеточный алгоритм,
- Жадный алгоритм.

Введем определение эффективности решения. Под эффективностью здесь понимается совокупность атрибутов решения: интерактивность, рациональное использование канала передачи данных (оптимальность сжатия), экономия трафика (минимизация данных на отправку).

Для построения эффективного решения требуется решить две алгоритмические проблемы. Первая – как эффективно покрыть изменения на экране, произошедшие за определенное время прямоугольниками пикселей. Критерий эффективности в данном случае – количество прямоугольников. Задача алгоритма – минимизировать это количество. Вторая проблема – как эффективно сжать эти прямоугольники перед отправкой на устройство клиента. В этом случае критерий эффективности – баланс между степенью сжатия и временем сжатия/распаковки.

Постановка алгоритмической задачи

В результате очередного обновления экрана перед серверным приложением ставится задача оптимального покрытия изменённых точек прямоугольниками, в общем случае известная как “задача о минимальном покрытии множества”. Формулируется она следующим образом. Пусть даны множество $M = \{1,...,m\}$ и набор его подмножеств $M_1,...,M_n$ таких,