Использование вероятностных сетей для автоматизации тестирования интернет приложений

Быков А. А.
Кафедра информатики
Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
e-mail: anton.bukov@gmail.com

***Аннотация* — В работе рассмотрена технология автоматизации тестирования интернет приложений, основанная на вероятностных сетях. Использование интеллектуальных эвристических алгоритмов в процессе автоматизации способна существенно упростить работу тестировщика, обеспечить гарантированное покрытие тестами всех элементов интерфейса тестируемого приложения.**

Ключевые слова: вероятностные сети; тестирование; web интерфейсы

I. Существующие технологии автоматизации тестирования

На сегодняшний день специализированные средства тестирования, предоставляют развитые средства для обработки результатов, полученных в процессе тестирования, однако используют интерпретируемые языки, обладающие слабой поддержкой ООП [1].

HP Mercury Quick Test Professional позволяет записывать скрипты, использует язык VBScript или JScript, а также централизованное хранилище GUI элементов. IBM Rational Functional Tester позволяет записывать скрипты тестирования на языке SQABasic, описывая элементы GUI в виде маппинга, в последствии можно написать алгоритм и обращаться к элементам динамически описывая элементы интерфейса на основе их параметров (класс, имя/текст). Borland SilkTest использует в своей работе язык 4Test, позволяет автоматически записывать тестовые скрипты, использует карты отображения GUI, структурно описывая интерфейс.

Это вынуждает программистов использовать объектно-ориентированные языки для автоматизации тестов и эмулятор действий пользователя. К примеру, средство автоматизации Silenium позволяет эмулировать действия пользователя, используя языки python, платформы .net и java [2]. Средство автоматизации AutoIt предоставляет собой ActiveX объект, и может быть использован любым языком, способным использовать ActiveX объекты.

II. Использование вероятностной сети для тестирования приложения

Достижение покрытия приложения тестами согласно выбранного критерия один из важнейших вопросов автоматизации тестирования. Сегодня существует несколько технологий, позволяющих определить степень покрытия исходного кода тестами по различным критериям, однако инструмента, способного определить процент покрытия интерфейса тестами согласно некоторого критерия черного ящика такого как: “тестирование функций”, «тестирование допустимых значений» не существует. Современные системы управления процессами контроля качества способны контролировать только критерий покрытия пунктов спецификации. Решить эту проблему можно создавая модель тестируемого интерфейса, используя Вероятностные сети.

Вероятностные сети – это система для принятия решений при условии противоречивости данных. Для реализации тестирования вероятностная сеть может иметь следующую структуру:

Сеть первого уровня, описывающая элементы интерфейса состоит из двух слоев. Узлы нижнего уровня представляют собой шаблоны для идентификации графических элементов, а узлы верхнего уровня – объекты, из которых состоит интерфейс. Сеть, описывающая интерфейс состоит из узлов, которые описывают графические элементы, используемые в интерфейсе программы, а связи – компоновку интерфейса. Цель данной сети определить расположение элементов управления на web странице.

Сеть второго уровня, описывает состояния программы и действия над её интерфейсом. Она состоит из двух видов узлов – узлов всех возможных состояний программы и узлов всех возможных действий над программой. Связи сети, описывают переходы между состояниями в результате этих действий. Главная задача сети – описать алгоритм работы программы в виде простых, связанных между собой правил.

Сеть третьего уровня, описывает тесты и дефекты тестируемой программы. Узлы нижнего слоя соединены с узлами-действиями сети второго уровня, верхний слой – описывает набор записанных тестов. Эти два слоя могут быть разделены ещё одним слоем, описывающим дефекты, которые были найдены в тестируемом приложении.

Сеть четвертого уровня, описывает знания о целях тестирования. Сеть состоит из узлов, каждый из которых представляет свою цель и связан с одним или несколькими тестами. Примером цели может быть как тестирование некоторой функции программы, так и такая сложная цель как поиск взаимного влияния компонентов программы.

III. Автоматизация при помощи вероятностных сетей

Сеть для тестирования приложения может создаваться, на основании записи действий тестера. Данный способ удобен, если система тестирования обладает скудными знаниями о тестируемом приложении. При записи теста система тестирования сохраняет последовательность состояний приложения и действий над интерфейсом. После записи теста тестеру предлагается ответить на некоторые вопросы, которые появились у системы тестирования. Результатом записи должна стать сеть-диаграмма переходов между состояниями.

Для результата (конечного состояния) теста заводится некоторая характеристика успешности. Впоследствии, для тестирования одного или нескольких модулей, система повторит записанные действия, используя алгоритм поиска пути в графе состояний, предложенным С. Расселом [3]. Наличие целей тестирования непосредственно соответствующих пунктам спецификации разрабатываемого приложения позволяет гибко изменять стратегию тестирования.

Ещё одной проблемой автоматизации тестирования является адаптация автоматизированных тестов к новым версиям тестируемого приложения [4]. Если система не уверена в правильной идентификации состояний или операций она последовательно будет задавать вопросы об их принадлежности новым или уже известным шаблонам состояний web интерфейса.

Если в процессе записи была выполнена неизвестная ранее комбинация действий, между известными состояниями, системе необходимо определить, является ли данная последовательность аналогичной уже известным системе комбинациям, при каких условиях можно выполнять записанные действия.

Наличие шаблонов графических элементов и состояний web интерфейса тестируемого приложения позволяет контролировать покрытие web интерфейса тестами, эффективно адаптировать записанные тесты к новым версиям тестируемого приложения.

Если аналогичная последовательность действий повторяется многократно, система может предложить объединить данную последовательность в подсеть (функцию). Объединение части сети в подсеть имеет значение только для удобства восприятия теста человеком. Стоит заметить, что записываемые действия и состояния не будут дублироваться. Записывая второй и последующие тесты, система добавляет только неизвестные состояния и операции. Если некоторые состояния были объединены в подсеть, это не помешает системе связывать их с состояниями, которые находятся вне этой подсети. Часто в процессе автоматизации неудачно выбранная декомпозиция теста на функции усложняет процесс автоматизации. Использование единой, целой модели тестируемого интерфейса позволяет избежать дублирования и выполнять рефакторинг записанных тестов.

Наконец, сеть для тестирования приложения может создаваться при помощи ответов на вопросы системы тестирования. Данный способ эффективен, когда система тестирования имеет достаточно знаний о тестируемой программе. При этом система тестирования будет в фоновом режиме выполнять тестирование приложения, и при, обнаружении проблемы, обращаться к тестировщику не останавливая выполнение остальных тестов.

Работа системы и тестировщика начинается с некоторого начального состояния приложения. Это состояние анализируется и, если состояние не соответствует представлениям системы, шаблонам состояния система задает вопросы. Для прощения диалога с пользователем все вопросы сводятся просто к подтверждению найденных изменений или, в случае ошибки, выбору верного решения. К примеру, если система тестирования надежно определяет основные элементы управления, система попросит просто подтвердить компоновку страницы. Далее система выбирает наиболее приоритетную для тестирования функцию. Далее система анализирует состояние, предлагает, в случае необходимости, создать характеристику описывающую дефект в программе.

Понятие дефекта или “бага” в программе, для сети, это характеристика, значение которой может изменить логику работы одной или нескольких связей, при этом значение характеристик успешности тестирования не изменяется. Жизненный цикл дефекта состоит из 4-х значений:

1) найден, но не описан
2) описан, но не исправлен
3) исправлен, но исправление не проверено
4) не воспроизводится

Для разработчика и тестировщика результат тестирования должен отображаться по-разному. Если тест окрашен красным, это означает, что для связанной с тестом операции должны быть выполнены соответствующие действия. Так для состояния дефекта 2 тест для тестировщиков будет зеленым, а для разработчиков красным. Для 3 – наоборот, для тестировщика красным, а для разработчика зеленым.

Наличие жизненного цикла у дефекта позволяет интегрировать систему учета дефектов и систему автоматизированного тестирования.

Таким образом, система автоматизации тестирования на основе Вероятностных сетей способна определить процент покрытия интерфейса тестами согласно критериев черного ящика, эффективно объединять записанные тесты в модель и контролировать воспроизводимость дефектов для новых версий тестируемого приложения.

[1] I. Vinnichenko Automation of testing processes, Peter Press, C-Piterburg, 2005.

[2] Alan Richardson, Selenium Simplified, A tutorial guide to using the Selenium API in Java with Junit, Compendium Developments, Great Britain, 2010

[3] S. Russell, P. Norvig, Artificial intelligence: a modern approach (AIMA), Williams, Moscow, 2007.

[4] V. V. Kulemin, Methods of software verification, Institute for System Programming, Moscow, 1995.