МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

ПРОБЛЕМИ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Рекомендовано до друку Вченою радою
Херсонського національного
технічного університету
(протокол №5 від 08 червня 2010 року)

Журнал «Проблеми інформаційних технологій»
включено до Переліку наукових фахових видань
ВАК України (Постанова Президії ВАК України
№1-05/5 від 21.05.2008 р.), у яких можуть
публікуватися результати дисертаційних робіт на
здобуття наукових ступенів доктора
та кандидата наук

ISSN 1998-7005

#01(007) червень 2010

ФОРМИРОВАНИЕ БАЗОВОГО НАБОРА ТЕСТОВЫХ СЦЕНАРИЕВ ДЛЯ WEB-БАЗИРОВАННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

УДК 004.415.532.2

КАЛИТА Надежда Ивановна
к.т.н., доцент кафедры Системотехники Харьковского национального университета радиоэлектроники.
Научные интересы:
информационные технологии разработки Web-базированных систем.

БАТУРИНА Марина Андреевна
студентка кафедры Системотехники Харьковского национального университета радиоэлектроники.
Научные интересы:
методы тестирования информационных систем.

ВВЕДЕНИЕ.

Процесс разработки Web-базированной информационной системы (WB ИС), как и любой другой ИС, состоит из ряда взаимосвязанных этапов. В рамках этого процесса осуществляются работы, начиная с анализа системы, определения её функций и требований к ней, и заканчивая её разработкой, внедрением и сопровождением. Неотъемлемой частью этого процесса является этап тестирования, позволяющий определить соответствие разработанной системы предъявляемым к ней требованиям [1].

Задача тестирования информационных систем (ИС) не нова, однако её методы не могут быть применены к Web-базированным ИС, поскольку их тестирование имеет ряд особенностей, обусловленных спецификой данных систем [2]. Такие системы представляют собой клиент-серверные приложения, доступ к которым осуществляется при помощи браузера, то есть посредством Web-интерфейса. Кроме того, ручное тестирование больших Web-приложений, которые имеют множество входных параметров, связанных с множеством записей в базе данных сложной структуры, малоэффективно. При разработке таких программных продуктов вопрос об автоматизации процесса тестирования стоит особенно остро.

ЦЕЛЬ СТАТЬИ.

Целью статьи является разработка нового метода тестирования Web-базированных информационных систем, эффективность которого обеспечивается за счет автоматической генерации базового набора тестовых сценариев на основе анализа информации о входных данных.

ОСНОВНОЙ МАТЕРИАЛ.

Тестирование Web-базированных ИС включает в себя проверку:

- функциональности;
- навигации (корректность ссылок, время загрузки

страниц);
- работоспособности системы как минимум в трёх наиболее часто использующихся браузерах (Internet Explorer, Opera, Firefox);
- удобства использования (логически правильные последовательности действий для пользователя);
- безопасности, которая включает аутентификацию пользователя, возможность доступа к страницам через url и защиту от «взлома» системы;
- релевантности полей данных: проверку обязательных для заполнения полей, диапазонов, округления значений, загрузки файлов, полей с возможностью выбора значений.

Проверка навигации, работоспособности системы в наиболее часто использующихся браузерах, удобство использования и проверка безопасности проводятся, как правило, в ручном режиме, поэтому выходят за рамки рассмотрения.

Для проверки соответствия разработанной системы функциональным требованиям применяются методы тестирования «черного ящика» (функциональное (поведенческое) тестирование, тестирование удобства использования, целостности данных, бизнес циклов, конфигурации) и «белого ящика» (модульное, комплексное тестирование, тестирование утечки памяти и обработки ошибок) [3].

На сегодняшний день при осуществлении функционального тестирования Web-базированных ИС широко применяются такие подходы, как: 1) тестирование вручную; 2) тестирование с использованием тестовых сценариев; 3) автоматизированное тестирование.

Проверка WB ИС вручную приводит к большим временным и трудовым затратам. Специалисту по тестированию необходимо изучить модуль, предназначенный для тестирования, составить ряд сценариев, охватывающих основные возможные поведения системы, а также вручную проверить корректность работы системы при выполнении этих сценариев. В совокупности такие действия занимают достаточно много времени и, кроме того, одним из недостатков данного подхода является человеческий фактор – специалист по тестированию может пропустить ошибки в работе системы в связи с усталостью и монотонностью работы.

В рамках второго подхода специалист по тестированию анализирует логику работы проверяемого модуля, на основании чего создает перечень ситуаций (тест-кейзов), которые в дальнейшем будут использованы для контроля работоспособности системы. Этот подход эффективнее предыдущего, так как в процессе тестирования создаётся и документируется чёткая последовательность действий для проверки модуля, однако один из основных недостатков – человеческий фактор – остаётся. Многократное повторение одних и тех же действий снижает уровень концентрации и внимания специалиста по тестированию, что влечёт за собой снижение качества проверки.

Тестирование с использованием автоматизированных средств [4] является наиболее эффективным, так как специалист по тестированию избавлен от рутинной работы по воспроизведению тестовых сценариев. Достоинством данного подхода также является возможность многократного выполнения определённой последовательности действий в ходе регрессионного тестирования.

Наибольшее распространение получили такие средства автоматизированного тестирования, как JUnit, iMacros и Selenium [5-7].

Возможности JUnit включают:
- наличие элементов Assertions, позволяющих проверить совпадение полученных результатов с ожидаемыми;
- возможность предварительной установки общих данных для всех тестов;
- вывод информации о ходе тестирования в текстовом или графическом виде.

Достоинства JUnit:
- является тестовой средой;
- не требует контроля со стороны пользователя во время исполнения;
- возможность запуска нескольких тестов одновременно;
- ведение журнала ошибок.

Недостатками JUnit являются:
- сообщение только о первой ошибке в тесте;
- запуск нескольких тестов одновременно не позволяет отследить момент их завершения;
- необходимость специализированных знаний.

Основное отличие и достоинство инструментов Selenium [6] состоит в том, что тесты выполняются в браузере, а не в приложении, эмулирующем его поведе-

ние. Для записи последовательности действий (скриптов) можно применять простой текстовый редактор либо воспользоваться средством Selenium IDE.

К достоинствам Selenium относятся:

- кроссплатформенное тестирование;
- тесты Selenium выполняются непосредственно в браузере;
- для автоматизации тестов можно использовать один из нескольких поддерживаемых языков программирования (Java, Ruby, Python, Perl, PHP);
- возможность записать определённую последовательность действий и многократно её воспроизводить;
- понятный скрипт;
- возможность задания скорости воспроизведения записанной последовательности действий;
- возможность определять параметры, по которым будет производиться идентификация элемента страницы: расположение, ссылка, id;
- возможность воспроизведения скрипта как с начала, так и с указанного места.

Недостатки Selenium:

- неработоспособность скрипта при изменении интерфейса;
- сложность в изменении скрипта.

iMacros [7] является расширением для браузера Firefox, и позволяет записывать определенные действия в макросы, а потом воспроизводить их. iMacros for Firefox позволяет заполнять формы, автоматизировать загрузку станиц и извлекать из них данные. iMacros сохраняет все данные форм в файле, шифруя их, может читать данные страницы и экспортировать их в формат CSV. При этом имеется полная поддержка Unicode, работа с любыми языками, включая китайский.

Достоинства iMacros:

- возможность записи последовательности действий и последующего её воспроизведения;
- понятный и легко редактируемый макрос;
- Недостатки iMacros:
- нестабильность в работе;
- работает только с браузером Mozilla Firefox;
- неработоспособность скрипта при изменении элементов интерфейса.

Анализ современных используемых подходов к тестированию (вручную, с применением тестовых сценариев и автоматизированное тестирование) показывает, что их существенными недостатками является 1) способность только записывать определённую последовательность действий и ее воспроизводить; 2) ориентация на конкретную Web-базированную ИС.

Постановка задачи.

Эффективность тестирования (E) зависит от двух критериев: количества проверок K и времени их проведения t.

$$E = F(K, t), \qquad (1)$$

где $F(K, t)$ – некоторая функция, отображающая зависимость эффективности от количества проверок и времени их проведения. При этом чем больше вариантов будет проверено, тем меньшая вероятность получения ошибки, то есть количество проверок должно быть максимальным:

$$K \to \max. \qquad (2)$$

Если учитывать только количество проверенных данных, то наиболее эффективное тестирование получается в результате полного перебора. Возможность ошибки исключается, но время проведения такого тестирования огромно.

С другой стороны, чем меньше время тестирования, тем быстрее будет получен результат и тем скорее можно приступать к тестированию следующего модуля, то есть:

$$t \to \min \qquad (3)$$

Таким образом, эффективность тестирования E является функцией противоречивых критериев, и для повышения эффективности тестирования необходимо найти наилучшее соотношение между количеством тестовых ситуаций и временем проведения тестирования.

Тестирование Web-ориентированных ИС включает в себя проверку функциональности (логики взаимодействия) и проверку релевантности (ограничений на исходные данные). Проверка функциональности включает в себя проверку логики работы модуля, правильность заполнения значений одних полей в зависимости

от значений других, а также проверку взаимосвязей тестируемого модуля с другими частями системы. Проверка внутренней логики работы относится к тестированию «белого ящика», при котором известны взаимосвязи и взаимозависимости между всеми полями, а также логика их взаимодействия.

Проверка ограничений на исходные данные (включающая в себя проверку заполнения полей, возможности ввода пустых строк, корректных и некорректных значений, проверку диапазона), как правило, относится к тестированию «черного ящика».

Анализ существующих методов и средств тестирования позволяет сделать вывод о том, что на сегодняшний день не существует такого средства тестирования, которое на основании информации об исходных данных может сформировать перечень тестовых сценариев для проверки соответствия требованиям любой Web-базированной ИС.

В связи с этим возникает задача разработки нового метода тестирования, который для любой Web-ориентированной ИС на основании исходных данных, включающих информацию о типах полей и ограничениях на их значения, автоматически генерирует базовый набор тестовых сценариев.

Требования к методу:

- генерация базового набора тестовых сценариев в автоматическом режиме;
- осуществление проверки релевантности исходных данных на основании информации о них, которая задается шаблоном в соответствии с типом данных;
- тестирование всех типов полей: чисел, дат, текстовых полей и полей с возможностью единичного или множественного выбора значения;
- универсальность, т. е. возможность применения к любой Web-базированной ИС.

РЕШЕНИЕ ЗАДАЧИ.

Метод формирования базового набора тестовых сценариев состоит из четырёх этапов:

1. Определение множества исходных данных, то есть перечня полей на тестируемой странице.
2. Подробное описание каждого исходного значения: отнесение его к одному из стандартных типов полей, указание ОДЗ, ограничений на значения, указание обязательности заполнения.
3. Составление множества тестовых сценариев на основании полученных данных.
4. Сохранение информации для каждого поля о проверенных значениях в его ОДЗ.

Рассмотрим каждый из этапов.

Этап 1. Определение множества исходных данных.

На первом этапе специалист по тестированию составляет список проверяемых полей на Web-странице (4):

$$IN = \{ in_1, in_2, \ldots, in_i, \ldots, in_k \}, \quad (4)$$

где in_i – множество полей на странице;

in_i – название i -го поля;

k – количество полей на странице.

Этап 2. Описание исходных данных.

На втором этапе специалистом по тестированию определяется список проверяемых полей на Web- странице с указанием их названия, типа и обязательности заполнения. Данные о полях записываются в следующем формате:

$$in_i = < id, type, obligatory >, \quad (5)$$

где id – идентификатор поля;

type – тип поля;

obligatory – обязательность заполнения поля: если поле является обязательным для заполнения, ставится значение «1», иначе – «0».

Все поля, находящиеся на странице, делятся на 4 типа: числа, даты, текстовые поля и поля с возможностью выбора значения. К последним относятся поля с возможностью единичного или множественного выбора. Формат записи информации о полях зависит от типа. Рассмотрим подробно каждый из них.

Для числового поля информация задаётся в формате:

$$in_i = <pos\_neg, wh\_frac, min\_num, max\_num, delta\_num, ODZ\_num >, \quad (6)$$

где pos_neg – знак числа. Если данное поле может содержать только положительные числа, при заполнении шаблона по данному полю записывается значение «pos», если только отрицательные – «neg», если и те, и другие – значение «all»;

wh_frac – целое или дробное значение. Если поле может содержать только целые числа, при заполне-

нии шаблона записывается значение «wh», если только дробные – «frac», если и те, и другие – значение «all»;

min_ num –минимальное значение поля;

max_ num –максимальное значение поля;

delta_num – шаг изменения значения поля;

ODZ_num – область допустимых значений (ОДЗ) данного поля.

Для поля, содержащего информацию о датах, описание задаётся в формате:

$$\dot{n}_i = <\text{format, min\_data, max\_data, delta\_data, ODZ\_data}>, \quad (7)$$

где format – формат даты (например, ДД.ММ.ГГГГ, ДД.ММ.ГГГГ ЧЧ:СС или ММ.ДД.ГГГГ);

min_data – минимальное значение поля;

max_data – максимальное значение поля;

delta_data – шаг изменения значения поля;

ODZ – область допустимых значений данного поля.

Для текстового поля описание задаётся в формате:

$$\dot{n}_i = <\text{min\_text, max\_text}>, \quad (8)$$

где min_text –минимальное значение поля;

max_text – максимальное значение поля.

Поля с возможностью выбора включают в себя поля с единичным и множественным выбором. Для описания этих типов полей используются соответственно форматы (9) и (10):

$$\dot{n}_i = <\text{qua\_list, val1, val2,}\ldots, \text{val}_{\text{qua\_list}}>, \quad (9)$$

$$\dot{n}_i = <\text{am\_list, qua\_ch, val\_list1, val\_list1, val\_list2,}\ldots, \text{val\_list}_{\text{am\_list}}>, \quad (10)$$

где qua_list, am_list – количество возможных значений в списке;

qua_ch – максимально возможное количество выбранных значений. Если могут быть выбраны все значения, в шаблоне указывается значение «all», иначе указывается максимально возможное количество выбранных значений;

val1, val2,..., valqua_list, val_list1, val_list2,..., val_listam_list – значения поля, возможные для выбора.

Этап 3. Составление тестовых сценариев.

Далее составляется базовый набор тестовых сценариев для каждого поля тестируемой страницы. Сценарии составляются на основании типа проверяемых полей с учетом их особенностей и свойств, которые отражены в описаниях (6)-(10). Такой подход позволяет избежать полного перебора вариантов тестовых сценариев, сократив их количество, и быстро выработать стратегию создания тесткейза, что удовлетворяет требования локальных критериев (2), (3).

Для каждого типа поля формируется набор тесткейзов. Все тесткейзы хранятся в базе данных (БД) тестовых сценариев. На основании информации о поле из БД сценариев для тестирования выбирается соответствующий тесткейз.

Тестовый сценарий представляется в формате:

$$T_j = a,\ j = \overline{1, m} \quad (11)$$

где T_j – тестовый сценарий для заданного типа поля;

m – общее количество сценариев;

a – значение проверяемого поля.

В тестовых сценариях указываются значения полей, удовлетворяющие заданной ОДЗ, граничные, а также неверные значения, при указании которых проверяемая система должна корректно себя вести и выдавать сообщения о неверном заполнении полей.

Тестовые сценарии по рассматриваемому методу разработаны для всех типов полей. Рассмотрим тестовые сценарии на примере типа поля «Число».

1. Числовое поле должно содержать только положительное число (значение параметра «pos_neg»= pose»):

$T_1 = <a_1>$,

где a_1 – положительное число, принадлежащее ОДЗ;

$T_2 = <a_2>$,

где $a_2 = \max - \text{delta\_number}$;

$T_3 = <a_3>$,

где $a_3 = \min + \text{delta\_number}$;

$T_4 = <a_4>$,

где $a_4 = \max$;

$T_5 = <a_5>$,

где $a_5 = \min$;

$T_6 = <a_6>$,

где $a_6 = \max + \text{delta\_number}$;

$T_7 =< a_7 >$,

где $a_7 = \min - \text{delta\_number}$;

$T_8 =< a_8 >$,

где a_8 – отрицательное число;

$T_9 =< a_9 >$,

где $a_9 = 0$;

$T_{10} =< a_{10} >$,

где a_{10} – символьное значение (не число);

2. Числовое поле должно содержать только отрицательное число (значение параметра «pos_neg»= «neg»):

T_1,

где a_1 – отрицательное число, принадлежащее ОДЗ;

$T_2 =< a_2 >$,

где $a_2 = \max - \text{delta\_number}$;

$T_3 =< a_3 >$,

где $a_3 = \min + \text{delta\_number}$;

$T_4 =< a_4 >$,

где $a_4 = \max$;

$T_5 =< a_5 >$,

где $a_5 = \min$;

$T_6 =< a_6 >$,

где $a_6 = \max + \text{delta\_number}$;

$T_7 =< a_7 >$,

где $a_7 = \min - \text{delta\_number}$;

$T_8 =< a_8 >$,

где a_8 – положительное число;

$T_9 =< a_9 >$,

где $a_9 = 0$;

$T_{10} =< a_{10} >$,

где a_{10} – символьное значение (не число).

3. Числовое поле может содержать и отрицательное, и положительное число (значение параметра «pos_neg» = «all»). В данном случае следует применить все сценарии для проверки положительных и отрицательных чисел ($T_1 - T_{10}$).
4. Числовое поле должно содержать только целое число (значение параметра «wh_frac» = «wh»):

$T_1 =< a_1 >$,

где a_1 – целое число, принадлежащее ОДЗ;

$T_2 =< a_2 >$,

где a_2 – дробное число.

$T_3 =< a_3 >$,

где $T_{3=<a_3>}$ – дробное число с нулевой дробной частью;

$T_4 =< a_4 >$,

где a_4 – целое число, не принадлежащее ОДЗ;

$T_5 =< a_5 >$,

где a_5 – дробное число, не принадлежащее ОДЗ;

$T_6 =< a_6 >$,

где $a_6 = 0$;

$T_7 =< a_7 >$,

где a_7 – символьное значение (не число);

5. Числовое поле должно содержать только дробное число (значение параметра «wh_frac» = «frac»):

$T_1 =< a_1 >$,

где a_1 – дробное число, принадлежащее ОДЗ;

T_2,

где a_2 – дробное число с одним знаком после разделителя целой и дробной части;

$T_3 =< a_3 >$,

где a_3 – дробное число с несколькими знаками после разделителя целой и дробной части;

$T_4 =< a_4 >$,

где a_4 – дробное число с нулевой дробной частью;

$T_5 =< a_5 >$,

где a_5 – целое число, принадлежащее ОДЗ;

$T_6 =< a_6 >$,

где a_6 – целое число, не принадлежащее ОДЗ;

$T_7 =< a_7 >$,

где a_7 – дробное число, не принадлежащее ОДЗ;

$T_8 =< a_8 >$,

где $a_8 = 0$;

$T_9 =< a_9 >$,

где a_9 – символьное значение (не число).

6. Числовое поле может содержать и целое, и дробное число (значение параметра «wh_frac» = «all»). В

данном случае следует применить все сценарии для целых чисел и $T_2 - T_6$ из сценариев для дробных чисел.

7. Числовое поле является обязательным для заполнения (значение параметра «obligatory» = «1»):

   $T_1 = < a_1 >$,

   где a_1 – число, то есть поле заполнено;

   $T_2 = < a_2 >$,

   где a_2 – отсутствие значения, то есть поле не заполнено;

   $T_3 = < a_3 >$,

   где $a_3 = 0$;

   $T_4 = < a_4 >$,

   где a_4 – символьное значение (не число).

8. Числовое поле является необязательным для заполнения (значение параметра «obligatory» = «0»):

   $T_1 = < a_1 >$,

   где a_1 – число, то есть поле заполнено;

   $T_2 = < a_2 >$,

   где a_2 – отсутствие значения, то есть поле не заполнено;

   $T_3 = < a_3 >$,

   где $a_3 = 0$;

   $T_4 = < a_4 >$,

   где a_4 – символьное значение (не число).

Аналогичным образом составляются тестовые сценарии и для остальных типов полей.

Этап 4. Сохранение информации о проверенных значениях.

На четвёртом этапе метода для каждого поля на его области допустимых значений отмечаются уже проверенные значения, что позволяет собирать информацию о проверенных сценариях и сформировать сценарии для дальнейшего тестирования, не используя повторно уже проверенные значения.

ВЫВОДЫ.

Разработанный метод формирования базового набора тестовых сценариев для Web-базированных информационных систем является частью информационной технологии тестирования, который позволяет для любой Web-ориентированной информационной системы автоматически сформировать набор тестовых сценариев для проверки релевантности исходных данных. Для генерации сценариев необходима информация о типах полей и ограничениях на их значения.

ЛИТЕРАТУРА:

1. Грекул В.И. Проектирование информационных систем /В.И. Грекул, Г.Н. Денищенко, Н.Л. Коровкина. – М.: Интернет-университет информационных технологий-ИНТУИТ, 2008. – 304 с.
2. Стотлемайер Д. Тестирование Web-приложений. Пер. с англ. – М.: Издательский дом «Кудиц-Образ», 2003. – 240 с.
3. Тамре Л. Введение в тестирование программного обеспечения. – М.: Вильямс, 2003. – 368 с.
4. Дастин Э. Автоматизированное тестирование программного обеспечения /Э. Дастин, Д. Рэшка, Д. Пол. – М.: ЛОРИ, 2003. – 590 с.
5. Tahchiev P., Leme T., Massol M., Gregory G. JUnit in Action 2nd edition. -Manning Publications, 2009. – 456 p.
6. Brown T., Gheorghiu G., Huggins J. An Introduction to Testing Web Applications with twill and Selenium. -O'Reilly Media, 2007. – 520 p.
7. iMacros Software Manual. iOpus Software GmbH, 2006 – 155 p.