# ИНФОРМАТИКА И ПРОГРАММИРОВАНИЕ

УДК 004.054

**А.М. Гудов**

## МЕТОД «ПРОЗРАЧНОЙ ЖУРНАЛИЗАЦИИ» ДЛЯ ОРГАНИЗАЦИИ ПРОЦЕССА ТЕСТИРОВАНИЯ WEB-ОРИЕНТИРОВАННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Рассматривается организация процесса тестирования информационной системы (ИС), обладающей распределенной архитектурой и web-ориентированным интерфейсом пользователя, с использованием метода прозрачной журнализации. Данная методика позволяет объединить интеграционное и системное тестирование ИС с учетом преимуществ стратегий «черного» и «белого» ящиков, автоматизировать процесс генерации тестовых сценариев, применять различные критерии для оценки полноты тестирования.

**Ключевые слова:** *процесс тестирования*, *интеграционное тестирование*, *системное тестирование*, *распределенная система.*

Тестированию приложений в последнее время уделяется очень много внимания. Существует огромное количество литературных источников [1 – 7], где приводятся рекомендации по проведению мероприятий по тестированию и верификации создаваемого программного обеспечения (ПО). В некоторых учебных заведениях даже открываются отдельные специальности, которые готовят специалистов в этой области – тестеров. Однако не существует универсального подхода или достаточно общих рекомендаций для проведения тестирования – в каждом конкретном случае тестер опирается на конкретное ПО, решающее конкретную задачу.

Многократно увеличивается сложность тестирования при создании систем с распределенной архитектурой – «база данных + сервер приложений + web-интерфейс». Процесс тестирования этих приложений требует дополнительных затрат на «интеграционное тестирование» (тестирование системного интерфейса) и «системное тестирование» (тестирование интерфейса пользователя). Поэтому разработка новых подходов, облегчающих этот процесс, по-прежнему является актуальной задачей перед «выходом» информационных систем. В статье предлагается один из подходов, названный автором «серым журналом», с помощью которого в настоящее время тестируются как отдельные приложения, так и целые подсистемы единой интегрированной аналитической информационной системы управления вузом, которая на протяжении нескольких лет разрабатывается в Кемеровском государственном университете.

### 1. Краткий анализ основных методов тестирования

Общая идеология тестирования ориентирована на использование двух основных подходов – тестирование в режиме «белого ящика» или «черного ящика». В первом случае процесс тестирования проводит разработчик системы, используя

исходный код. Во втором – тестируется уже готовое приложение или (его модули) с опорой только на исполняемый код системы. Существует промежуточный подход, который использует идеи первых двух – тестирование «серого ящика», где тестеру частично или полностью предоставлен исходный текст системы, а также ее исполняемый код. Такой подход позволяет наиболее качественно провести все процедуры тестирования ПО.

Однако при тестировании реальных приложений для определения и реализации полного набора тестовых примеров может оказаться недостаточно времени или других ресурсов. В такой ситуации нужно определить, какие тесты наиболее важны, а какие функции анализируемого ПО не будут тестироваться.

Как правило, тестер в реальных условиях придерживается подхода тестирования в режиме «черного ящика». Далее кратко проанализированы две наиболее распространенные методики такого тестирования – «метод наращиваемого подхода» [1 – 4] и «схематический подход» [5]. Обе методики соответствуют функциональному тестированию ПО.

### 1.1. Метод наращиваемого подхода

Метод наращиваемого подхода (подробно изложен в [3]) применяется в том случае, когда тестер, имея исполняемый код системы, проходит последовательно несколько стадий тестирования.

*Изучение приложения и ознакомление с его функциями* – это необходимый этап обучения. Тестеры создают тесты на ходу, часто находясь под влиянием предыдущих тестов. Этот подход, известный как исследовательское тестирование, – первый этап в процессе принятия решения о том, что будет тестироваться.

Недостатки: результаты, выданные приложением, могут повлиять на решение тестера – он может поверить, что полученные результаты верны; когда тестер сталкивается с ошибкой, может оказаться, что для определения верного пути в приложении или предыдущего состояния нет записи последовательности входных параметров, действий пользователя и т.д. Это может усложнить воспроизведение ошибки.

*Базовое тестирование* – создание базового тестового примера. Обычно базовый тест несложен: он основывается на простейшем пути в приложении, в процессе которого используются установки по умолчанию или же другие непосредственные входные данные. При заданных входных данных тестер должен определить результирующие данные (любые наблюдаемые выходные данные или отсутствия таковых).

Недостатки: отсутствие точного представления о правильных данных; выходные данные не могут гарантировать правильность прохождения даже базового теста.

*Анализ тенденций* – это необязательная стадия, на которой проводится отслеживание характера поведения путем изменения значения одной определенной переменной. Даже если результат реальных выходных данных точно не известен, можно оценить, изменяются ли значения выходных данных в ожидаемом направлении.

*Инвентаризация* – состоит из определения типов данных, доступных в приложении с последующей классификацией состояний, в которых может находиться каждый элемент данных. Тестирование позволяет убедиться, что каждое состояние используется по крайней мере один раз. Каждый набор состояний задает инвентарный список.

Недостатки: после того как разработчики устранят проблему и создадут новую версию приложения, тестеры должны повторно провести исходные тесты.

*Комбинирование элементов инвентарных списков*, чтобы можно было определить, предсказуема ли их совместная работа. Не все инвентарные списки содержат простые значения данных или состояния.

Недостатки: при определении всех комбинаций получается огромное число тестовых примеров – значительно больше, чем можно было бы выполнить в разумные сроки.

*Граничные оценки* – исследование пределов возможностей приложения. Пределы возможностей приложения (или границы) определяются данными. Примерами могут служить минимальные и максимальные значения диапазона данных; минимальный и максимальный размер поля и т.д. Общее правило – создать три тестовых примера, чтобы охватить следующие значения: граничное значение; граничное значение –1; граничное значение +1.

*Ошибочные данные* – попытка вывести приложение из строя, создавая условия, в которых обычный пользователь, вероятнее всего, работать не будет. Цель заключается в проверке приемлемого поведения приложения даже в том случае, если тестовые данные не описывают нормального состояния. При создании тестового примера с ошибочными данными в результате обычно появляется сообщение об ошибке. Ожидается, что приложение отловит эти данные и проинформирует пользователя об ошибке.

*Создание напряжений*. На этой стадии тестеры отходят от функций приложения и оценивают условия их реализации в терминах рабочих характеристик и восстановления системы после неполадок. Даже если система перезагружается, после перезапуска все должно функционировать корректно.

**Резюме:** Метод наращиваемого подхода (эволюционный подход) – применим как основа для тестирования любого приложения. Однако метод не учитывает особенности приложения. Метод поддается «автоматизации» на стадии, когда известны необходимые тестовые задания (сценарии тестирования). Труден при проведении интеграционного тестирования за счет того, что необходимо «охватить» все возможные комбинации (записи инвентарных списков) входных параметров и получаемых состояний с «прицелом» на совместное функционирование всех модулей системы. Требует много времени для «всеохватывающего» тестирования приложения.

### 1.2. Схематический подход

Схематический подход – это метод, который используется для рассмотрения требований [5]. Требования преобразуются в схемы для того, чтобы перечислить различные условия теста. Схематический метод управляет процессами и не зависит от содержания системы. Нередко он позволяет обнаружить отсутствующую в требованиях информацию.

Каждый элемент в схеме, в конечном счете, будет определять входные состояния для тестового примера. Набор состояний системы может быть трансформирован в набор функций и т.д. Схема, как правило, представляет собой древовидную структуру (граф), в которой между корнем и каждым листом существует однозначно определяемый путь. Этот путь устанавливает специфический набор входных состояний, которые затем будут использованы для определения тестового примера.

Схема содержит узловые точки (или точки ветвления), обозначающие подсказки, тип значения или входной вариант. Нумерация узла обозначает специфическое входное состояние соответствующего ему предка. Число листьев на дереве (или путей в схеме) дает приблизительное число тестовых примеров, необходимых для тестирования всех функций приложения.

Создание схемы – итерационный процесс. Последняя версия схемы помогает окончательно сформулировать тестовые сценарии. Каждый путь в схеме (от корня к листьям) определяет входную комбинацию, которая образует основу для тестового сценария.

Процесс тестирования разбивается на последовательные стадии и поддерживает различные уровни [6,7]: поэлементное тестирование (затрагивает наименьшие единицы компиляции какого-либо приложения); тестирование уровня интеграции (заключается в комбинировании отдельных элементов и подтверждении того, что они функционируют корректно); тестирование уровней системы (применяется для целостных приложений, чаще всего касается поведения пользователя при работе с системой). Результаты тестирования записываются в таблицы. На основании этих данных делаются оценки о покрытии тестами структуры приложения.

**Резюме:** Таблицы применяются на протяжении всего процесса тестирования, начиная с поэлементного тестирования и заканчивая тестированием на уровне системы. Описание приложения при помощи диаграммы переходов несложно преобразовать в эквивалентную таблицу состояний, из которой потом легко выводятся тестовые сценарии. Для приложений, где используется большое количество комбинаций сложных данных, в таблицах тестов могут быть сделаны ссылки на файлы, которые содержат реальное описание входных данных и исходных результатов. С помощью таблиц решений можно отслеживать сложные комбинации условий и связанные с ними результирующие действия.

### 1.3. Тестирование web-приложений и тестирование баз данных

Приложения, основанные на технологиях WWW, несут в себе новые проблемы как для разработки, так и для тестирования [4, 8, 9]. Тесты для web-узла должны быть ориентированы на предполагаемое поведение узла. Необходимо оценивать следующие проблемные моменты: функциональные возможности; практичность; навигацию; форму, содержимое страницы. Подробно принципы тестирования web-приложения изложены в [4].

Тестирование баз данных часто является очень важной частью тестирования приложения [10]. При тестировании баз данных требуются всесторонние знания тестируемого приложения. К ключевым проблемам, возникающим при тестировании баз данных, относятся целостность данных; достоверность данных (подходящая форма при вводе в базы данных); манипуляции с данными и обновления.

**Общее резюме:** Тестирование распределенного приложения довольно сложная задача. Если приложение использует многоуровневую архитектуру «клиент – сервер», то описанные выше подходы должны применяться для каждого уровня. Но при этом требуется провести интеграционное тестирование для проверки системных интерфейсов и системное тестирование для проверки внешних интерфейсов (как правило, интерфейсов пользователя или внешней системы).

Обычно для сложных приложений тестирование распадается на инкрементальное – тестирование отдельных компонент, отдельных модулей, отдельных

подсистем, отдельных частей приложения; интеграционное – тестирование взаимодействие всех частей приложения, интерфейса пользователя, интерфейса системы в целом.

Общей методики не существует, поскольку нужно использовать «особенности» приложения. Эффективного тестирования невозможно достигнуть, не зная исходного кода (структуры) всех компонент системы. Особенно задача осложняется в том случае, если система разрабатывается отдельными частями или в систему интегрируются «чужие» модули.

## 2. Интеграционное тестирование распределенной системы

Для практической демонстрации предлагаемого подхода используем реальную информационную систему, имеющую следующую распределенную архитектуру (рис. 1): клиент через web-браузер обращается к приложению; запрос обрабатывает сервер приложений, при необходимости вставляет данные, получаемые по запросу с сервера баз данных, формирует выходной HTML-файл; сформированная страница отправляется клиенту.

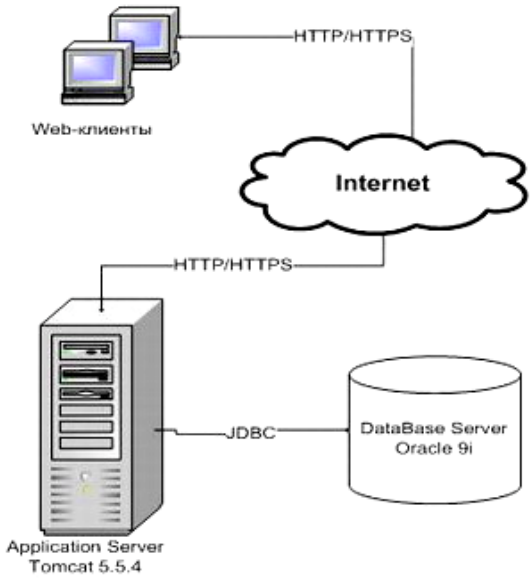


Рис. 1. Архитектура распределенной системы

На сервере приложений обработкой данных занимается пакет KemsuWEB, структура которого показана на рис. 2. Основным носителем информации является XML-файл шаблона будущей страницы с данными. В шаблоне присутствуют специальные конструкции, предназначенные для расширения возможностей обработки данных языка XML: вызов процедур или выполнение запросов для получения данных с сервера базы данных; управление логическими инструкциями; обработка локальных переменных и переменных сессии (глобальных для данного сеанса пользователя).

Другими словами, сервер приложений разделяет уровни хранения данных, обработки данных и формирования интерфейса пользователя (рис. 3).

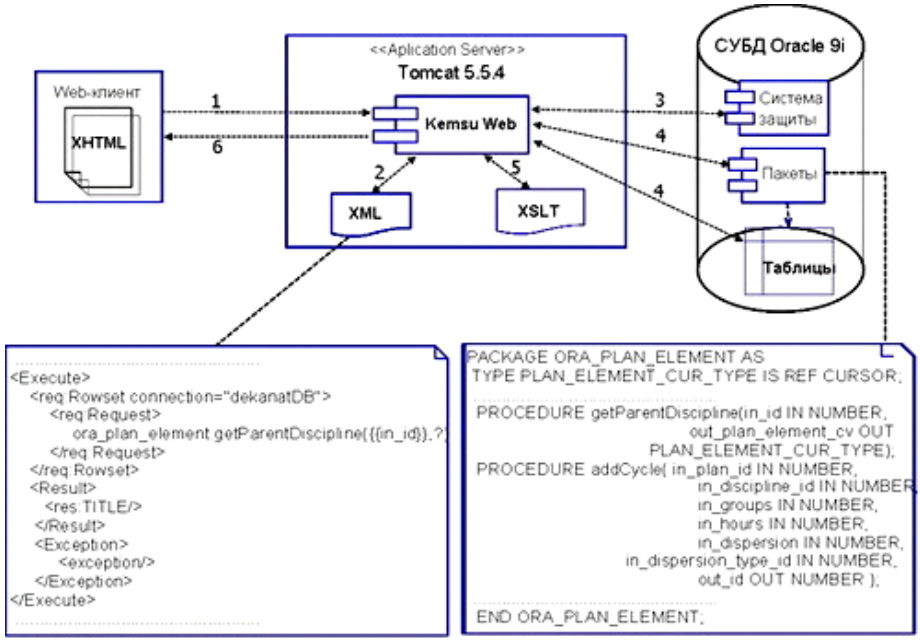


Рис. 2. Архитектура пакета KemsuWEB


Рис. 3. Уровни работы приложения

Процесс тестирования такого приложения представляет собой сложную задачу. Для её решения мало провести инкрементальное тестирование отдельных компонент приложения, интерфейса пользователя, корректности работы запросов к данным и т.д. Необходимо накопить статистику выполнения тестовых сценариев и состояний приложения, определить структуру тестируемой системы. Кроме того, при выполнении тестовых сценариев желательно иметь возможность быстро изменять структуру приложения и определять ветку структурного графа приложения, которую необходимо проверить. Множество значений входных параметров, задаваемых через формы web-интерфейса, также необходимо накапливать вместе с ответными сообщениями системы.

Для проведения тестовых мероприятий предлагается методика «прозрачной журнализации», которая ниже будет пояснена на примере тестирования лишь небольшой части системы – компонента авторизации на сервере конференций КемГУ (http://conference.kemsu.ru) (рис. 4).

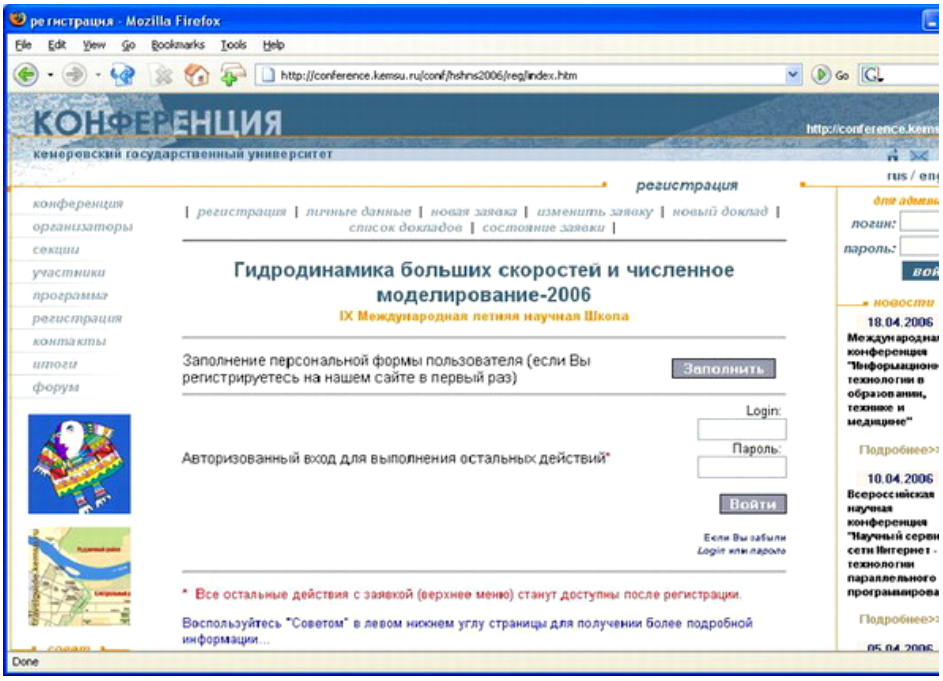


Рис. 4. Пример web-приложения для иллюстрации методики тестирования

На рис. 5 приведен фрагмент исходного кода интерфейса пользователя, реализованного в соответствующем XML-файле.

```
<center><table style="font-size: 11pt">
[1]  <res:If name="$$PersonID$$" value="" op="eq">
     <res:Then>
          <tr>
          <td>Заполнение персональной формы ...</td>
[2]       <form action="conf/{{conferenceAlias}}/reg/ins_person.htm" method="post">
          <input type="hidden" name="in_lang" value="RU"/>
          <td align="center" colspan="2"><input type="submit" class="btn" value="Заполнить"/></td>
          </form>
          </tr>
          <tr>
          <td>Авторизованный вход для выполнения остальных действий</td>
[3]       <form action="conf/{{conferenceAlias}}/reg/chk.htm" method="post">
          <td align="right" style="font-size: 10pt">
               Login: <input type="text" name="in_login" value="" size="10"/><br/>
               Пароль: <input type="password" name="in_passwd" size="10"/><br/><br/>
               <input type="submit" class="btn" value="Войти"/>
          </td>
          </form>
          </tr>
</res:Then>
<res:Else>
          <tr>
[4]       <th align="left">Авторизованный вход для </th>
          <td align="right" style="font-size: 10pt" colspan="2">
               <i>##LAST_NAME## ##FIRST_NAME## ##MIDDLE_NAME##</i>
               <br/>##ORGANISATION##<br/>##CITY##<br/>##COUNTRY##
          </td>
</res:Else>
 </res:If>
</table></center>
```

Рис. 5. Фрагмент исходного кода, реализующего форму авторизации пользователя

Поскольку этот файл имеет текстовый формат и всегда доступен для просмотра/редактирования (в рамках доступных привилегий системы защиты), то его использование дает тестеру ряд преимуществ при ознакомлении и организации графа структуры web-приложения.

Для иллюстрации предлагаемой методики используем только часть формы (рис. 4), определяющую поведение системы в двух случаях:

- если пользователь впервые зашел на сайт конференции, ему необходимо ввести свои регистрационные данные (нажать кнопку «заполнить»);
- если пользователь уже имеет личные данные в этой системе, ему необходимо выполнить авторизованный вход для дальнейшей работы (заполнить поля формы «логин» и «пароль», нажать кнопку «войти»).

## 3. Метод прозрачной журнализации

Суть метода заключается в следующем. В исходном XML-файле тестер расставляет специальным образом метки (на рис. 5 они приводятся в квадратных скобках). Далее при выполнении этого кода метки вместе с фрагментами исходного кода и значениями соответствующих переменных помещаются в специальную таблицу – журнал активности. Формат записи в простейшем случае приведен в табл. 1.

Таблица 1

**Формат записи журнала активности**

| Имя модуля (компонента) | Метка узла | Метка след. узла | Действие /предикат | Значение предиката | Возвращаемое значение | Комментарий |
|---|---|---|---|---|---|---|
| | | | | | | |

Каждая запись представляет собой элемент графа структуры тестируемого приложения. Набор записей полностью отображает таблицу на граф. Фрагмент графа структуры приведен на рис. 6, а соответствующий ему набор записей в журнале активности – в табл. 2.

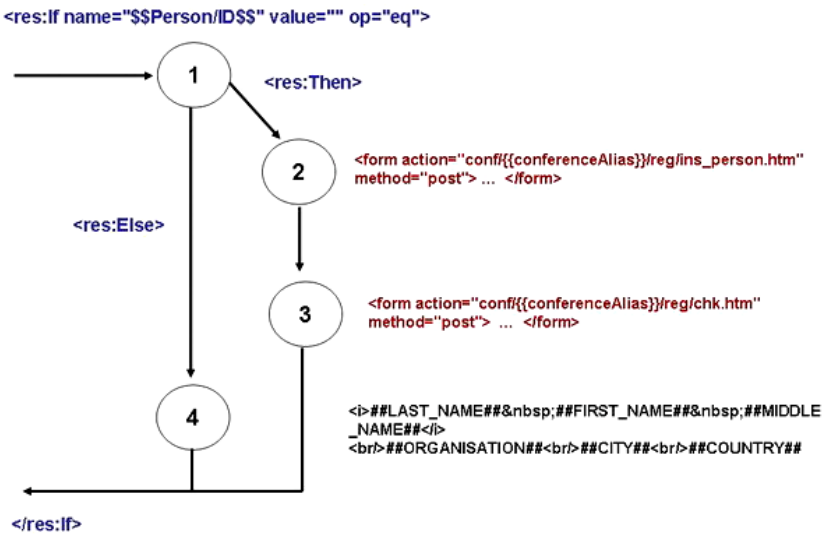


Рис. 6. Фрагмент графа структуры приложения

Таблица 2

**Фрагмент записей журнала активности для рассматриваемого примера**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| … | | | | | | | |
| 1 | http://conference.kemsu.ru/conf/reg/index.htm | [1] | [2] | If name ="$$Person/ID$$" value="" op="eq" | true | | Значение **Person/ ID = null** |
| 2 | http://conference.kemsu.ru/conf/reg/index.htm | [2] | [3] | conf/{{conferenceAlias}}/reg/ins_person.htm | Значения передаваемых параметров | | Передача управления форме |
| 3 | http://conference.kemsu.ru/conf/reg/index.htm | [3] | … | conf/{{conferenceAlias}}/reg/chk.htm | Значения передаваемых параметров | Значение выходных параметров | Передача управления форме |
| 4 | http://conference.kemsu.ru/conf/reg/index.htm | [1] | [4] | ##LAST_NAME## ##FIRST_NAME## ##MIDDLE_NAME## ##ORGANISATION## ##CITY## ##COUNTRY## | Значения переменных | Откуда вернулись | Вывод в XML-файл |
| … | | | | | | | |

В строке таблицы с номером 1 проверяется состояние авторизации пользователя (узел с номером 1 на рис. 6). Если значение переменной $$Person/ID$$ неопределенно, то пользователь не прошел регистрацию в системе и ему необходимо заполнить свои данные при помощи специальной формы (узел с номером 4 на рис. 6).

Если пользователь получил свои данные для авторизации (логин и пароль), ввел их в соответствующие поля формы и начал процесс авторизации (кнопка «войти» на рис. 4), его параметры передаются в специальный файл «ins_person. htm» (узел с номером 2 на рис. 6), где происходит проверка его учетных данных (узел с номером 3 на рис. 6, строки 2 и 3 в табл. 2). При этом значения передаваемых параметров и результат процесса авторизации помещаются в столбец 6 табл. 2. Эти данные в последующем могут служить для автоматической генерации тестовых сценариев при проведении повторного или более детального тестирования. В случае успешной проверки учетной записи пользователь переходит на свой рабочий стол и продолжает работу с системой. Если попытка авторизации прошла неудачно, пользователь остается в пределах текущей формы.

Если пользователь начал процесс регистрации в системе (узел с номером 4 на рис. 6) посредством специальной формы, то его параметры могут быть помещены в столбец 5 табл. 2 (переменные #LAST_NAME##, ##FIRST_NAME##, ##MIDDLE _NAME##, ##ORGANISATION##, ##CITY##, ##COUNTRY##) и могут быть использованы для проведения детального тестирования процесса регистрации.

После выполнения необходимых тестовых сценариев тестер имеет в своем распоряжении структуру проверяемой части модуля (компонента) системы, набор значений передаваемых/принимаемых переменных, набор сообщений системы (помещаются в последнее поле) в случае возникновения проблемных ситуаций. На основании этих данных можно провести анализ покрытия структуры приложения. Анализ покрытия дает возможность оценить эффективность теста (с исполь-

зованием специализированных инструментов тестирования) и провести наблюдение за путями выполняемого кода.

Журнал активности сохраняется в специальной схеме базы данных, где и происходит накопление необходимых статистических данных. Написав несложный набор утилит, тестер получает мощный инструментарий для последующего анализа работы приложения.

## Заключение

Предлагаемая методика объединяет на своей основе стратегии интеграционного тестирования (тестирование модулей/компонент в едином приложении с использованием подходов, присущих функциональному тестированию, т.е. методика «серого ящика») и системного тестирования (тестирование внешних интерфейсов или интерфейсов пользователя).

Записи из журнала можно использовать сколько угодно раз. Совокупность записей можно конвертировать в любой формат представления, который можно использовать при других методах автоматизации процесса тестирования, например с помощью сетей Петри.

Записи журнала «прозрачны» для различных компонент приложения (на чтение/запись).

На основании записей из журнала невозможно провести тесты, подтверждающие корректность содержимого web-страницы с точки зрения пользователя. Это нужно сделать другим способом.

ЛИТЕРАТУРА

1. *Beizer B.* Software Testing Techniques (2nd edition). Van Nostrand Rienhold Company, 1990.
2. *Beizer B.* Black Box Testing. John Wiley, 1995.
3. *Dumas J.S.*, *Redish J.* A Practical Guide to Usability Testing (revised edition). Ablex, 1999.
4. *Тампе Л.* Введение в тестирование программного обеспечения: пер. с англ. М.: Изд. дом «Вильямс», 2003.
5. *Блейзер Б.* Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем. СПб.: Питер, 2004.
6. *Myers G.J.* The Art of Software Testing. John Wiley, 1976.
7. *Kaner C.*, *Falk J.*, *Nguyen H.Q.* Testing Computer Software (2nd edition). John Wiley, 1999.
8. *Nguyen H.Q.* Testing Applications on the Web: Test Planning for Internet-Based Systems. John Wiley, 2001.
9. *Splaine S.*, *Jaskiel S.P.*, *Savoia A.* The Web Testing Handbook // Software Quality Engineering. 2001.
10. *Dustin E.R.*, *RashkaJ.*, *Paul J.* Automated Software Testing: Introduction, Management and Perfomance. Addison-Wesley, 1999.

Статья представлена оргкомитетом VII Всероссийской научно-практической конференции с международным участием «Информационные технологии и математическое моделирование».

*Гудов Александр Михайлович*
Кемеровский государственный университет
E-mail: good@kemsu.ru

Поступила в редакцию 19 января 2009 г.