УДК 004.054

Т.И. Брагина, аспирант,
Г.В. Табунщик, к. т.н., доц.,
Запорожский национальный технический университет, г. Запорожье, Украина
Bragina.zntu@gmail.com, galina.tabunshchik@gmail.com

Стратегия тестирования web - проектов

В статье рассмотрено место web-проектов среди информационных систем и определены ключевые моменты их тестирования. Предложена стратегия тестирования web-проектов, основанная на анализе структуры прототипа.

Ключевые слова: управление качеством, web-проект, прототип, методы тестирования.

Введение

Качество web-проекта формируется из качества самого продукта, как степени соответствия требованиям пользователей, и качества процесса разработки. Одним из эффективных инструментов контроля качества является тестирование, которое необходимо применять на всех этапах жизненного цикла (ЖЦ) проекта.

Однако web-проектам присущи свои специфические проблемы. Во-первых, данные и требования к web-проектам часто меняются, хранятся в различных структурированных и неструктурированных форматах в системе каталогов, что создает проблему отслеживания их актуальности и корректности. Во-вторых, тесты для web-проекта должны быть спроектированы так, чтобы они были готовыми к возможному изменению требований, применяемых технологий и общей архитектуры. Все эти обстоятельства требуют хорошо организованного процесса управления качеством, которое должно закладываться на этапах анализа и спецификации требований к web-приложению и подтверждаться на всех этапах ЖЦ [1].

Набор методов тестирования зависит от специфики структуры и назначения web-проектов, но, в основном, является результатом личного опыта тестировщика. Поэтому актуальным является исследование видов web-проектов с целью определения их особенностей и специфических инструментов для их тестирования.

Постановка задачи

Сегодня существует огромный выбор разнообразных методов и средств тестирования для различных видов информационных систем (ИС) и различных задач.

Основной вклад в исследование проблемы тестирования внесли:

- Д. Грахам [2] – определил основы тестирования для получения сертификата International Software Testing Qualifications Board – ISTQB;

- Т. Ландауэр, Я. Нильсен [3] – вывели эмпирические формулы для определения наилучшей стратегии тестирования удобства пользования;

- Л. Криспин, Д. Грегори [4]– исследовали гибкое тестирование;

- С. В. Синицын, Н.Ю. Налютин [5] – исследовали процесс верификация программного обеспечения;

- Б. Бейзер [6] – исследовал поведенческое тестирование разнообразных систем, направленное на поиск методов, позволяющих отыскать максимум ошибок при минимуме временных затрат;

- Ю.А. Скобцов, В.Ю. Скобцов [7]– исследовали генерацию тестов для современных компьютерных систем.

Их исследования затрагивают различные стороны тестирования программного обеспечения и доказывают необходимость выработки стратегии тестирования и внедрения методик тестирования на ранних этапах развития проекта.

Для того, чтобы определить методы тестирования для web-проектов, выделим их место среди ИС.

Чаще всего ИС классифицируют по следующим признакам [8]:

- по степени распределённости (настольные или локальные ИС, распределённые ИС (файл-серверные, клиент-серверные));

- по степени автоматизации (автоматизированные, автоматические);

- по характеру обработки данных (информационно-справочные, или информационно-поисковые ИС; ИС обработки данных, или решающие ИС);

- по сфере применения (экономическая ИС, медицинская ИС, географическая информационная система и т.д.);

- по охвату задач (масштабности) (персональная ИС, групповая ИС, корпоративная ИС).

На более общем уровне все ИС можно разделить на 2 типа – небольшие и крупномасштабные проекты. Небольшие проекты разрабатываются фрилансерами или независимыми командами разработчиков, в то время как крупномасштабные разрабатываются группой специалистов с проведением анализа концепции разработки и функционированием всех процессов по управлению проектами.

Небольшие проекты в свою очередь разделим по целям (коммерческие и некоммерческие) и по назначению (системы переводов, независимые компоненты web-проектов, поисковая оптимизация (search engine optimization – SEO), приложения для мобильных платформ (iPhone, Android, Symbian)). Крупномасштабные проекты по сфере применения разделим на автоматизированные системы управления (АСУ), проблемно-ориентированные web-проекты и ИС на базе компонент (Service-Oriented Architecture (SOA) и Enterprise Resource Planning (ERP) системы). Разработанная классификация представлена на рисунке 1.

Как видно из предложенной выше классификации, независимые компоненты web-проектов могут использоваться в различных видах ИС и быть включенными в крупные web-проекты. Поэтому выделим наиболее распространенные классы web-проектов [9]:

1. Системы электронной коммерции:
   1.1 Web-витрины;
   1.2 Интернет-магазины;
   1.3 Торговые Интернет системы;
   1.4 Интернет системы снабжения;
   1.5 Торговые ряды;
   1.6 Аукционы;
   1.7 Торговые площадки (marketplace);
2. Системы электронного ведение бизнеса:
   2.1 Электронное ведение бизнеса внутри корпорации;
   2.2 Электронное ведение бизнеса между партнерами;
   2.3 e-CRM-системы (Customer Relationship Management);
   2.4 Корпоративные порталы;
3. Контентные проекты:
   3.1 Каталоги;
   3.2 Горизонтальные порталы;
   3.3 Контентные проекты и вертикальные порталы;
   3.3 Online средства массовой информации (СМИ);
4. Социальные сети;
5. Поисковые системы;
6. Публичные порталы (Yahoo, Lycos, Excite, Rambler);
7. Развлекательные проекты (форумы, чаты, фотогалереи, flash-приложения).

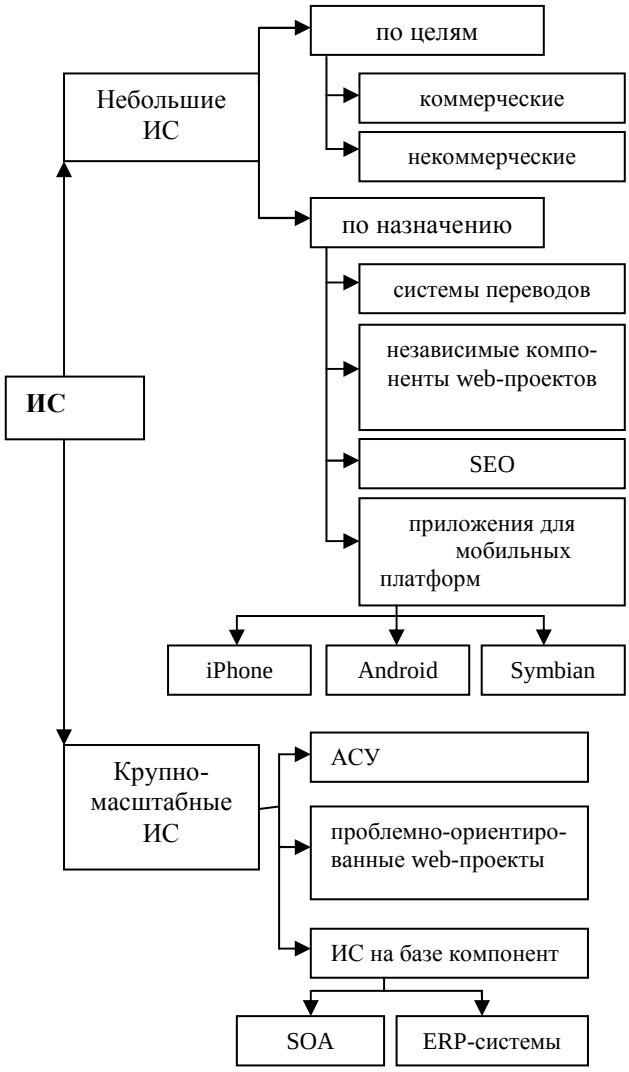

Рисунок 1 – Классификация ИС

Общими чертами для всех классов web-проектов является web-интерфейс, который позволяет формировать страницы в режиме интерактивного взаимодействия в системах «клиент-сервер».

Различие заключается в том, что критичность уровня качества web-проекта для крупномасштабных проектов несравнимо выше, чем для небольших ИС. Это объясняется тем, что выявление дефектов на этапе внедрения проекта вызывает необходимость возвращать проект на предыдущую стадию разработки, что существенно увеличивает затраты и нарушает график реализации.

Наибольшее внимание должно уделяться поиску ошибок в начале проекта, так как именно на начальных этапах принимаются наиболее важные решения. К тому же, по мере развития проекта, на каждом следующем этапе стоимость исправления дефекта снижается. Несмотря на это,

начальный уровень контроля качества является минимальным и повышается по мере продвижения процесса разработки, а полноценное тестирование начинается только на поздних этапах. Результатом этого часто является слишком позднее выявление самых дорогих ошибок и последующая дорогостоящая переработка системы либо ее отдельных частей.

В результате анализа вышесказанного можно сделать вывод о необходимости разработки метода комплексного тестирования web-проекта, ориентированного на контроле качества, начиная с этапа анализа требований, и отслеживающего изменения на каждом этапе ЖЦ.

Комплексный метод тестирования web-проекта

Комплексный метод тестирования заключается в использовании необходимых средств тестирования на каждом этапе ЖЦ проекта, начиная с создания прототипа будущего проекта. Предлагаемый метод подразумевает создание эволюционного прототипа, т.е. последовательно создается макет web-проекта, который будет с каждой итерацией ближе к реальному продукту. Такой подход имеет то преимущество, что на каждом шаге разработчики располагают работающей системой, которая обладает частичной функциональностью, которую можно тестировать и улучшать с каждой итерацией.

Для оценки соответствия прототипу предложено тестирование следующих метрик:

1. M_{N2} – соответствие количества страниц второго уровня вложенности версии проекта прототипу:

$$M_{N2} = \frac{N2}{N2_p} \times 100\%, \quad (1)$$

где *N2* – количество страниц версии проекта второго уровня вложенности;

$N2_p$ –количество страниц прототипа второго уровня вложенности;

2. M_{N3} – соответствие количества страниц третьего уровня вложенности версии проекта прототипу:

$$M_{N3} = \frac{N3}{N3_p} \times 100\%, \quad (2)$$

где *N3* – количество страниц версии проекта третьего уровня вложенности;

$N3_p$ – количество страниц прототипа третьего уровня вложенности;

3. $M_{N\to\infty}$ – количество страниц глубже третьего уровня вложенности:

$$M_{N\to\infty} = \frac{1}{N2_p + N3_p} \times \sum_{i=4}^{q} N_i \times 100\%, \quad (3)$$

где q – максимальный уровень вложенности версии проекта;

$N2_p$ и $N3_p$ – соответственно количество страниц прототипа второго и третьего уровня вложенности;

для лучшей индексации web-проекта $\lim M_{N\to\infty} \to 0$, т.е. количество страниц более чем третьего уровня вложенности должно быть как возможно меньшим по сравнению с количеством *N3* и *N2*.

4. M_S – соответствие структуры проекта прототипу (из-за симметричности относительно главной диагонали рассматривается только нижняя треугольная матрица):

$$M_s = \frac{\sum_{i=1}^{N}\sum_{j=1}^{i-1} a[i][j]}{N^2 - \sum_{i=1}^{N} i} \quad (4)$$

где N – количество страниц проекта и прототипа (в случае количество страниц не совпадает, добавляются недостающие строка и столбец и заполняются нулями);

$$a[i][j] = \begin{cases} 1, если\ S_p[i][j] = S[i][j] \\ 0, если\ S_p[i][j] \neq S[i][j] \end{cases},$$

S – матрица связей между страницами проекта;

S_p – матрица связей между страницами прототипа.

В заголовках строк и столбцов матрицы S_p находятся названия всех страниц прототипа web-проекта:

$$S_p[0][i] = S_p[i][0] = name\_page_i,$$

где $i \in [1; N]$ – номер страницы,

$name\_page_i$ – название *i*-той страницы.

Все элементы главной диагонали $S_p[i][i]=0.$ Остальные элементы матрицы $S_p[i][j]$ могут иметь следующие значения:

$$S_p[i][j] = \begin{cases} 0, \text{если связи между i - той и} \\ \quad j \text{ - той страницами отсутствуют;} \\ 1, \text{если существует возможность} \\ \quad \text{перехода с i - той на j - ую страницу;} \\ 2, \text{если существует возможность} \\ \quad \text{перехода с j - той на i - тую страницу;} \\ 3, \text{если существует двухсторонняя} \\ \quad \text{возможность перехода.} \end{cases} \quad (5)$$

Матрица *S* строится аналогично.

Данные метрики будут проверяться на каждом этапе ЖЦ проекта с помощью автоматизированного тестирования ссылок проекта. Это позволит проверять соответствие количества страниц всех уровней требуемому. Соответствие же содержания страниц и его качества оценивается заказчиком либо руководителем проекта (на промежуточных этапах разработки). Таким образом, с помощью этих метрик в методе определены средства, тестирующие на соответствие требованиям заказчика созданные на каждом этапе артефакты.

Ключевыми артефактами для web-проекта в зависимости от этапа ЖЦ являются:

- на этапе анализа требований – прототип интерфейса;
- на этапе проектирования – реакция отклика базы данных (БД);
- на этапе реализации – сервисы взаимодействия между слоями приложения.

В основу метода положен прототип интерфейса проекта, на основании которого вычисляются метрики оценки качества (1) – (4), связывающие логическую модель разрабатываемого проекта с артефактами на различных этапах создания проекта.

Рассмотрим применение метода на различных этапах ЖЦ проекта.

1. На **этапе анализа и спецификации требований** тестировщику, как одному из членов команды разработки, необходимо определить, являются ли собранные требования неясными, неполными, неоднозначными, или противоречащими. Процесс выявления и анализа требований в рамках разработанного метода состоит из двух параллельных процессов: создание матрицы трассировки требований и создание прототипа пользовательского интерфейса.

1.1. Матрица трассировки требований строится на основании выделенных требований заинтересованных лиц, функциональных и нефункциональных требований с помощью case-средств управления требованиями, например, Rational Requisite Pro [10]. После обработки полученных данных получаем таблицу со следующими обозначениями: STRQ – требования заинтересованных лиц, FEAT – функциональные требования, SUPL – нефункциональные требования, A – матрица трассировки, где «-1» – трассировка «trace to», «1» – трассировка «trace from».

В дальнейшем, с учетом направленности данного метода на анализ структуры, основное внимание будет уделяться функциональным требованиям и их связи с будущей структурой проекта. Матрица трассировки будет служить источником и способом отслеживания изменения требований, а соответственно и функциональности проекта.

Создание матрицы направлено на:

- подтверждение, что реализация соответствует требованиям (вся необходимая функциональность была реализована и протестирована);
- подтверждение, что вся реализована функциональность была назначена заказчиками;
- помощь в управлении изменениями (анализировать последствия изменения требований).

К атрибутам FEAT требований, таким как критичность, рискованность, срочность и так далее, для дальнейшего тестирования структуры проекта и использования выделенных требований для создания прототипа добавляется атрибут «Реализовать на странице». В данном поле необходимо указать название страницы, на которой планируется реализовать данную функциональность.

1.2. Для анализа требований к интерфейсу будущего проекта строится прототип пользовательского интерфейса. Применение прототипа при анализе и документировании требований позволяет решить проблемы понимания требований и дальнейшего моделирования и проектирования программной системы. При использовании прототипа для анализа требований возможен плавный и последовательный переход к моделям проектирования.

Параллельная работа по созданию матрицы и прототипа объединяется для отслеживания корректности структуры проекта – проверяется связанность функциональности и структуры страниц web-проекта. Для этого используется информация из атрибута FEAT требований «Реализовать на странице» и структура разработанного прототипа, полученная с помощью анализа прототипа.

Из поля «Реализовать на странице» извлекаются названия страниц, убираются повторяющиеся и из них составляет примерная структура будущего проекта:

$$attributes(FEAT) \Rightarrow structure_0(prototype),$$

где *structure₀ (prototype)* – начальная структура проекта, не содержащая информации об уровне вложенности страниц.

Затем автоматизируется процесс составления структуры прототипа за счет анализа внутренних ссылок. В зависимости от количества переходов от главной страницы определяется количество страниц со вторым N2p и третьим N3p уровнем вложенности. Страницы с уровнем вложенности более трех с точки зрения оптимизации являются недопустимыми при разработке структуры сайтов, так как поисковые роботы редко проводят их индексацию. Поэтому страницы, расположенные более чем в двух кликах от главной, рекомендуются к последующему дополнительному анализу и уменьшению уровня вложенности путем изменения структуры.

Затем определяется структура взаимодействия страниц – строится матрица возможных переходов между ними $S_p[N+1][N+1]$ (5).

В результате обработки на выходе разработчики получают *structure₁ (prototype)* – полноценную структуру прототипа в виде матрицы $St[N][2]$, в которой $St[i][1]$ – название *i*-ой страницы и $St[i][2]$ – уровень ее вложенности, определенный автоматически на основании связей, записанных в S_p.

Формально полученные на предыдущих шагах данные должны соответствовать друг другу:

$$structure_0(prototype) + S_p \sim$$
$$\sim structure_1(prototype)$$

Это соответствие проверяется и, в случае несовпадений, разработчики получают информацию о страницах прототипа, не несущих функциональной нагрузки, или о функциональных требованиях, ссылающихся на нереализованные в прототипе страницы. Применение данного алгоритма облегчает работу специалистов по выявлению требований и позволяет учесть и реализовать в прототипе все пожелания заказчика.

После установления соответствия структуры прототипа планируемой функциональности, прототип можно считать эталоном для создаваемого проекта, так как все последующие действия разработчиков будут направлены на создание web-проекта по прототипу, утвержденному заказчиком.

В процессе дальнейшей разработки, если страница проекта пройдет валидацию тестировщиками, заказчиком и руководителем проекта, всем FEAT требованиям, отнесенным к этой странице, расставится метка «Выполнено». Таким образом, можно будет отслеживать прогресс выполнения проекта по реализации функциональных требований.

2. На этапе проектирования web-проекта определяется архитектура классов проекта и проектируется БД, с которой будет взаимодействовать система. На первой итерации данного этапа предлагается автоматизировать заполнение информации о структуре проекта. Для этого используются данные, полученные при тестировании прототипа интерфейса web-проекта, т.е. структура прототипа записывается в БД как структура будущего проекта:

$$structure_0(project) = structure_1(prototype)$$

Во время итерационных возвратов на стадию проектирования либо в случае изменения прототипа тестируется соответствие структуры, записанной в БД, и полученной в результате анализа прототипа. Для этого используются метрики, записанные в формулах (1-4). Необходимо, чтобы $\lim M_{N2,3} \to 100\%$, $\lim M_S \to 100\%$ и $\lim M_{N\to\infty} \to 0\%$, поэтому при других значениях необходимо определить, из-за чего произошло отклонение от нормы, и исправить несоответствие прототипу или неактуальный прототип.

Применение метода тестирования БД на этапе проектирования позволяет:

1. Сократить время на первичное заполнение структуры web-проекта;
2. Определить несоответствия артефактов на первом и втором этапе разработки web-проекта.

Реализация проекта проходит итеративно, т.е. к этапу реализации возвращаются некоторое количество раз, что зависит от результатов тестирования проекта и изменения требований к нему. В разработанном методе на этапе реализации проекта классические методы тестирования в определенной последовательности объединяются с новыми предложенными методами контроля структуры проекта.

Первым этапом является дымовое тестирование новой версии проекта, по результатам которого делается вывод о возможности и целесообразности дальнейшего тестирования. В рамках данного метода особенное внимание при дымовом тестировании уделяют анализу ссылок – проверяется наличие пустых или некорректных ссылок. После их исправления или исключения их проекта, необходимо перейти к **этапу тестирования**, а

именно – централизованному тестированию по тестовым случаям (для первой версии проекта) и затем регрессионному тестированию для всех последующих версий.

С целью уменьшить количество тестовых ситуаций и объема тестируемого кода, необходимо определить какие структурные и функциональные части подвергнуть тестированию. Для решения данной проблемы авторами предлагается два действия:

1. Оценка соответствия разработанной версии и структуры, описанной в прототипе;

2. Определение частей, связанных с измененной областью, за счет структуры связей между страницами web-проекта.

Определение измененных частей проекта происходит в процессе санитарного тестирования как одного из элементов регрессионного тестирования, а именно путем отслеживания статусов дефектов. Чаще всего используются следующие статусы: новый, в разработке, исправленный, проверен исправлен, закрыт исправлен.

При выборе тестов для регрессионного тестирования первоначально для перепроверки выбираются области, которые содержат дефекты, которые разработчики перевели в статус «исправленный». Затем, ориентируясь на структуру проекта, определяются области, связанные с протестированной, и проверяются тестовые сценарии, составленные для этих областей. Таким образом, выявляется большая часть ошибок, появившихся в результате внесения изменений.

Формально данный процесс выглядит следующим образом:

$$if\ (fixed(bug_i) \cap bug_i \in FEAT_j \cap FEAT_j \in P_g) \Rightarrow$$

$$\Rightarrow test(P_g),$$

где bug_i – любой из незакрытых дефектов;

$FEAT_j$ – функциональное требование, к которому относится bug_i;

P_g – страница, к которой относится $FEAT_j$.

$$if\ (P_g \to P_k \cup P_g \leftarrow P_k) \Rightarrow test(P_k),$$

где $P_g \to P_k$ – переход от страницы P_g к P_k, $P_g \leftarrow P_k$ – наоборот.

$$if\ (test(P_g) = passed \cap test(P_k) = passed) \Rightarrow$$

$$\Rightarrow verified\ \ fixed(bug_i) \cap fixed\ close(bug_i),$$

где $verified\ \ fixed(bug_i)$ – присвоение bug_i статуса «проверен исправлен»;

$fixed\ close(bug_i)$ – присвоение bug_i статуса «закрыт исправлен».

Подобный процесс проходит от версии к версии, до тех пор, пока результаты тестирования не сойдутся с прописанными в плане тестирования критериями окончания тестирования. Работа, связанная непосредственно с тестированием, заканчивается тестированием безопасности. В успешных проектах метрики $M_{N2,3} = 100\%$, $\lim M_S = 100\%$ и $\lim M_{N\infty} \to 0$. Также их уровень может устанавливаться заказчиком.

Предложенный метод выходит за рамки стандартных целей контролировать качество и позволяет обеспечить качество на различных этапах ЖЦ. Это возможно за счет планомерного контроля и отслеживания соответствия разрабатываемого проекта требованиям заказчика к функциональности и структуре проекта. Созданные на каждом этапе артефакты тестируются на соответствие прототипу и, таким образом, данный метод предоставляет средства оценки качества на различных этапах создания проекта за счет анализа соответствия логической модели разрабатываемого проекта, принятой заказчиком.

Данный метод автоматизирован путем применения утилит, разработанных авторами. Для их работы используется информация из атрибутов FEAT требований, прототип и версий проекта. Она позволяет упростить процесс анализа структуры прототипа и проекта, отследить связи с функциональными требованиями, определить отсутствие переходов между страницами, проконтролировать соответствие проекта и прототипа, а также контролировать прогресс разработки проекта.

Заключение

Использование предложенного комплексного метода позволяет уменьшить количество ошибок на поздних этапах разработки проекта, т.к. за счет контроля структуры разрабатываемого проекта на каждом этапе ЖЦ разработки, ошибки определяются раньше, чем при использовании обычного процесса разработки, в котором тестирование происходит уже на завершающих стадиях.

Научная новизна заключается в разработке метрик оценки качества web–проекта, особенность которых состоит в эффективности их применения с различными методами тестирования. Практическая ценность данной

работы заключается в автоматизации процесса сбора информации и повышении эффективности тестирования за счет анализа структуры прототипа и web-проекта.

Список использованной литературы

1. Брагина Т.И. Средства обеспечения качества для моделей итеративной разработки программного обеспечения / Т.И. Брагина, Г.В. Табунщик // Зб. тез доповідей щорічної наук.-практ. конф. викладачів, науковців, молодих учених, аспірантів, студентів ЗНТУ «Тиждень науки», 12–16 квітня, 2010, Запоріжжя, Україна. – Запоріжжя, 2010. – С. 141 – 143.

2. Foundations of Software Testing: ISTQB Certification / D. Graham, E. van Veenendaal, I. Evans et al. // CENGAGE Lrng Business Press. – 2006. – 258 p.

3. Nielsen J. (1993). A mathematical model of the finding of usability problems / J. Nielsen, T. Landauer // Proc. ACM INTERCHI'93 Conf., 24-29 April, 1993, Amsterdam, the Netherland. – 1993. – P. 206-213.

4. Криспин Л. Гибкое тестирование. Практическое руководство для тестировщиков ПО и гибких команд / Л. Криспин, Д. Грегори // Вильямс. – 2010. – 464 с.

5. Синицын С.В. Верификация программного обеспечения / С.В. Синицын, Н.Ю. Налютин // Бином. Лаборатория знаний, Интернет-университет информационных технологий. – 2008. – 368 с.

6. Бейзер Б. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем / Б. Бейзер // Питер. – 2004. – 320 с.

7. Скобцов Ю.А. Построение тестов для перекрестных неисправностей типа задержка / Ю.А. Скобцов, В.Ю. Скобцов, Нассер Іяд К.М. // Наукові праці ДонНТУ. – 2011. – №14. – С .146-150.

8. Новомлинский Л. Интернет-стратегии каждый выбирает по себе: [Электрон. ресурс]. – Режим доступа: http://www.e-commerce.ru/analytics/analytics-part/analytics13.html

9. PMBOK® Guide and Standards: [Электрон. ресурс]. – Режим доступа: http://www.pmi.org/PMBOK-Guide-and-Standards.aspx.

10. Новичков А. Автоматизация процесса тестирования при помощи методологии и инструментальных средств IBM Rational. Часть 3.: [Электрон. ресурс]. – Режим доступа: http://software-testing.ru/library/vendors/154----------ibm-rational--3-

Надійшла до редколегії 30.03.2012

Т.І. БРАГІНА, Г.В. ТАБУНЩИК
Запорізький національний технічний університет, м. Запоріжжя, Україна

T.I. BRAGINA, G.V. TABUNSHCHYK
Zaporizhzhe National Technical University, Ukraine

СТРАТЕГІЯ ТЕСТУВАННЯ WEB – ПРОЕКТІВ

У статті розглянуто місце web-проектів серед інформаційних систем та визначено ключові моменти їх тестування. Запропоновано стратегію тестування web-проектів, засновану на аналізі структури прототипу.

Ключові слова: керування якістю, web-проект, прототип, методи тестування.

THE WEB - PROJECTS TESTING STRATEGY

The web-projects place in the information systems was considered and the key moments in their testing was identified in the article. The web-projects testing strategy, based on an analysis of the prototype structure, was proposed by the authors.

Keywords: quality management, web-design, prototype, test methods.