УДК 371.694:004:658.336:656.7.(045)

[1]**В. М. Синеглазов**, д-р техн. наук, проф.,
[2]**К. А. Хоменко**, студ.

ТЕСТИРОВАНИЕ ХРАНИМЫХ ПРОЦЕДУР ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ВЕБОРИЕНТИРОВАННЫХ ФУНКЦИОНАЛЬНЫХ ТРЕНАЖЕРОВ

[1]НТУУ «КПИ», e-mail: hkostya@gmail.com,
[2]Институт аэрокосмических систем управления НАУ, email: svm@nau.edu.ua

Предложено механизм тестирования хранимых процедур функциональных тренажеров для повышения отказоустойчивости систем.

Ключевые слова: модульное тестирование, хранимая процедура, функциональный тренажер, отказоустойчивость.

Вступление. С развитием аппаратной части появляется все больше возможностей для построения реалистичного интерактивного программного обеспечения для тестирования и обучения студентов. На сегодня существует ряд примеров интерактивных обучающих систем, которые в полной мере имитируют реальную среду, например, тренажер вождения автомобиля, полета на самолете, управления беспилотным летательным аппаратом и др. Большую популярность обрело онлайн обучение, где интерактивное тестирование как никогда актуально. В 2011 г. web сайт www.coursera.org, совместно с университетами США, запустил онлайн обучение, на курсы которого одновременно записалось более 70 000 студентов. Онлайн обучение такого масштаба выдвигает особые требования к отказоустойчивости программного обеспечение. Отказоустойчивость системы обеспечивается как конфигурационно с добавлением нового кластера, сервера, повышением производительных мощностей облака, так и на этапе разработки программного продукта оптимизация кода, способность выполнения кода в нескольких параллельных потоках, оптимизация базы данных.

Анализ существующих подходов к обеспечению качества программных продуктов. Существует множество подходов к обеспечению качества программного продукта. Рассмотрим следующие определения:

– [1061-1998 IEEE Standard for Software Quality Metrics Methodology]. Качество программного обеспечения – это степень обладания программным обеспечением требуемой комбинацией свойств;

– [ISO 8402:1994 Quality management and quality assurance]. Качество программного обеспечения – это совокупность характеристик программного обеспечения, относящихся к его способности удовлетворять установленные и предполагаемые потребности.

За последние десять лет большую популярность обрело автоматизированное тестирование или модельное тестирование (Unit testing, Test driven development). Автоматизированное тестирование программного обеспечения – часть процесса тестирования на этапе контроля качества в процессе разработки программного обеспечения. Оно использует программные средства для выполнения тестов и проверки результатов выполнения, что помогает сократить время тестирования и упростить его процесс.

В автоматическом тестировании выделяют четыре основных типа:

– модельное тестирование (Unit testing) – позволяет проверить на корректность отдельные модули исходного кода программы;

– интеграционное тестирование (Integration testing) – одна из фаз тестирования программного обеспечения, при котором отдельные программные модули объединяются и тестируются в группе. Обычно интеграционное тестирование проводится после модульного тестирования и предшествует системному тестированию;

– тестирование базы данных (Data testing) – процесс тестирования структуры базы данных, а также хранимых процедур или функций;

– тестирование производительности (Performance testing) – процесс тестирования программного обеспечения на удовлетворение требованиям производительности.

Большую популярность обрели именно модульные тесты, поскольку они просты в реализации и в значительной мере влияют на качество кода в целом. Стоит отметить, что модульные тесты не могут гарантировать отсутствия ошибок интеграции, которые сложно обнаруживаются. В случае частого возникновения таких ошибок и большого количества модулей, в разрабатываемой системе стоит внедрить интеграционное тестирование. Недостатком интеграционных тестов, в свою очередь, является сложность их внедрения в процесс беспрерывного развертывания. Зачастую такая интеграция влечет за собой рутину конфигурации среды и поддерживание ее в актуальном состоянии на протяжении всех итераций разработки. Data-тестирование встречается реже, поскольку требует значительных временных затрат на разработку и поддержку. Тестирование производительности не тривиальное по своей природе и актуально только в сильно нагруженных приложениях.

При разработке программного обеспечения модульное тестирование играет ключевую роль в обеспечении должного уровня качества и выполнения контрольных задач. Хотя многие разработчики согласны с тем, что модульные тесты важны, однак не уделяют должного времени написанию модульных тестов в конце цикла разработки продукта. Для решения этой проблемы и была задумана методология разработки через тестирование (TDD). Использующие эту методологию разработчики пишут модульные тесты для отдельных компонентов до их реального применения. Этот процесс позволяет разработчику «вывернуть наизнанку» интерфейс API и рассмотреть его внутренние процессы, что позволяет упростить проект настолько, насколько возможно.

Постановка задачи. Сформируем алгоритм тестирования хранимых процедур по методологии TDD на примере написания автоматических модульных тестов для одного из модулей системы интерактивного тренажера.

Анализ существующих подходов к тестированию хранимых процедур. Чтобы быть полезным, модульное тестирование должно устанавливать собственную среду выполнения перед применением, чтобы иметь возможность последовательно проверять код, который следует протестировать. После того как тест реализован и проверен, он регистрируется в системе контроля исходного кода, и его результаты используются в течении остатка жизненного цикла продукта, чтобы обеспечить требуемое поведение метода.

Применять TDD при разработке баз данных сложнее, поскольку для этого необходима единообразная тестовая среда, в которой код будет проверяться перед регистрацией изменений. Для успешного выполнения модульного теста базы данных он должен иметь соответствующую схему и подходящие данные. Это значит, что для выполнения тестов другим разработчикам придется воспроизводить среду, использовавшуюся во время создания модульных тестов. Для этого понадобится построение дорогостоящей инфраструктуры.

На данный момент распространены два подхода к тестированию хранимых процедур: тестирование путем написания еще одной хранимой процедуры, которая тестирует уже существующую; написание интеграционного теста, который использует уровень приложения для вызова тестируемого кода базы данных.

Первый подход менее популярен по ряду причин:

– временной ресурс, необходимый на написание тестов, часто превышает время, необходимое на написание тестируемой процедуры;

– отсутствие специализированной среды для запуска таких тестов;

– сложность использования в процессе беспрерывной интеграции;

– сложность поддержания таких тестов.

Второй подход модульного тестирования баз данных отражает две отчасти конкурирующие цели: стремление разработчика базы данных использовать известные сценарии T-SQL для выражения логики модульного теста и желание воспользоваться широкими возможностями языков, предназначенных для Microsoft® .NET Framework, таких как C# или Visual Basic. Противоречие между этим целями разрешается путем инкапсулирования теста в код T-SQL и затем размещения T-SQL в коде C# или Visual Basic.NET.

В стеке разработки на платформе .NET распространены следующие инструменты, призванные упростить написание модульных тестов для базы данных:

– DBPro – это среда разработки (IDE) от Microsoft, предоставляющая широкий спектр функций для создания модельных тестов. Среди преимуществ стоит отметить: создание модульных тестов-заглушек T-SQL из существующих функций, хранимых процедур или триггеров, автоматическое развертывание обновлений проекта базы данных в локальном экземпляре «песочницы», создание данных в качестве части установки тестовой среды с использованием плана автоматического создания данных и исполнение тестов T-SQL на целевой базе данных;

– NDbUnit –.NET библиотека для управления состоянием базы во время выполнения тестов. NDbUnit очень простой в использовании и бесплатный в отличие от DBPro.

Тестирование хранимой процедуры интерактивного тренажера. Рассмотрим цикл написания хранимой процедуры, одного из модулей обучающей системы, по методологии TDD.

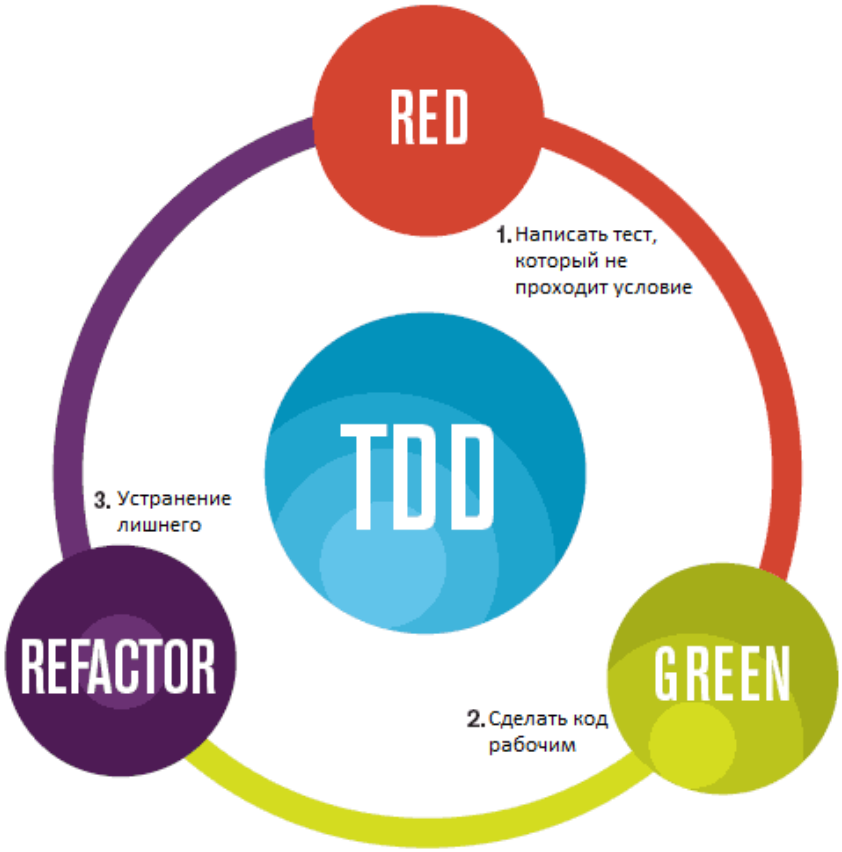

Рис. 1. Разработка посредством тестирования «red, green, refactor»

На первом этапе пишется код, который проверяет еще не существующий модуль на удовлетворение требованиям спецификации. На втором этапе разрабатывается сам модуль таким образом, чтобы он удовлетворял требованиям уже написанных тестов. И на третем (последнем) этапе проводится рефакторинг кода.

Требования к разрабатываемому модулю. Хранимая процедура должна возвращать среднюю оценку студента за указанный период в зависимости от полученных параметров. Алгоритм подсчета изображен на рис. 2.

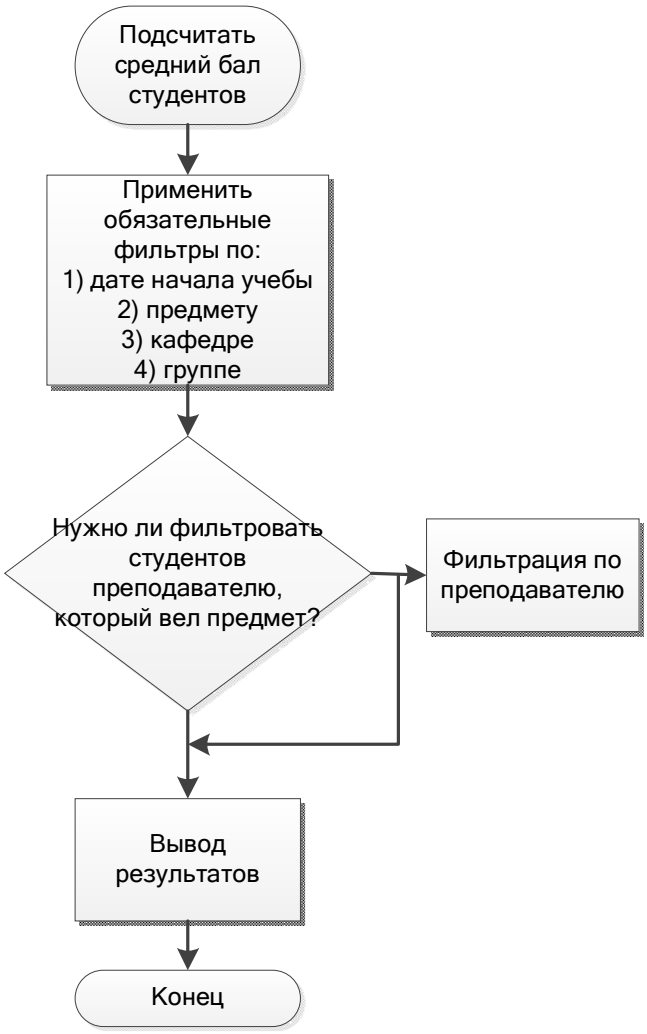

Рис. 2. Алгоритм подсчета среднего бала студентов

Согласно алгоритму на первом этапе необходимо написать шесть тестов:

1. Проверка правильности фильтрации по дате начала учебы.
2. Проверка фильтрации по предмету.
3. Проверка фильтрации по кафедре.
4. Проверка фильтрации по группе.
5. Проверка на то, что если флаг для фильтрования студентов по преподавателю, который вел предмет, установлен, то должна проводиться соответствующая фильтрация.
6. Проверка, что фильтрация не выполняется, если соответствующий флаг отключен

Структура тестового проекта будет иметь следующий вид (рис. 3):

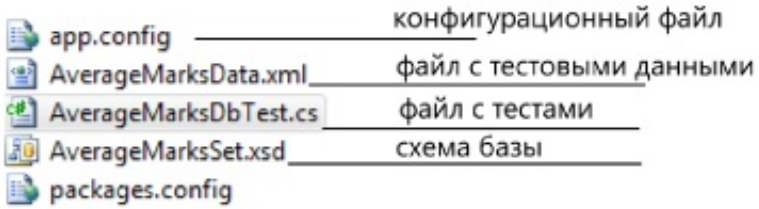

Рис. 3. Структура тестового проекта

После запуска тестов убеждаемся в том, что еще нет нужной хранимой процедуры для удовлетворения описанных правил.

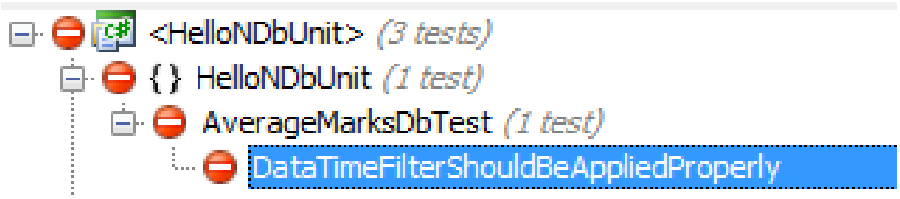

Рис. 4. Отчет выполнения модульных тестов

На втором этапе реализуется нужная хранимая процедура. Критерием завершения реализации – прохождение всех тестов, написанных в первом этапе.

Последний этап подразумевает рефакторинг хранимой процедуры, в то же время тесты помогут отслеживать большинство ошибок.

Вывод. Модульное тестирование стало неотъемлемой частью гибкой методологии разработки, а тестирование хранимых процедур – это еще один шаг усовершенствования процесса тестирования. В большинстве случаем наиболее удобным подходом к обеспечению такого тестирования является написание интеграционных тестов на уровне аппликации.

Список литературы

1. *Osherove Roy* The Art of Unit Testing: With Examples in .Net / Roy Osherove – Manning Publications, 2009. – 320 p.
2. *Лафлин Джейми* – Применение разработки через тестирование к проектам баз данных / Джейми Лафлин/ MSDN Magazine – 2008. – Режим доступа: msdn.microsoft.com/ru-ru/magazine/cc164243.aspx.
3. *Тарасов С.* SQL и модульное тестирование / С. Тарасов // Мир ПК. – 2008. – №7 – С. 32 – 48.
4. *Краковецкий* А. Microsoft User Group – Модульное тестирование ASP.NET приложений с использованием Moles и Pex / А. Краковецкий – 2010, С. 1 – 4.
5. *Бек Кент* Test Driven Development: By Example / Кент Бек / Addison-Wesley Professional. – 2002. – 240 p.

В. М. Синєглазов, К. А. Хоменко

Тестування збережених процедур програмного забезпечення веборієнтованих функціональних тренажерів

Запропоновано механізм тестування збережуваних процедур функціональних тренажерів для підвищення відмовостійкості системи.

V. M. Sineglazov, K. A. Khomenko

Unit testing stored procedures of the web-oriented simulator

This article defines the approach of unit testing stored for the interactive simulator in order to improve fault tolerance of the system.