УДК 004.052

В.А. КОВРИГИН, А.В. БОЯРЧУК

Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Украина

АНАЛИЗ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ТЕСТИРОВАНИЯ WEB-СЕРВИСОВ

Рассмотрены аспекты тестирования web-сервисов, выделена специфика написания и применения web-сервисов, обусловливающая особенности при тестировании. Проведен анализ методов и технологий тестирования, отдельное внимание уделено модульному тестированию. Проанализированы имеющиеся на данный момент инструментальные средства, применяющиеся для тестирования web-сервисов – Web Service – Fault Injection Technology (WS-FIT) и System Under Benchmarking (SUB) and Benchmark Management System (BMS).

web-сервис, надежность, тестирование, модульное тестирование, инструментальные средства

Введение

Проблема тестирования web-сервисов. В связи с все возрастающей популярности сервер ориентированной архитектуры (англ. Server Oriented Architecture, SOA), остро встает вопрос обеспечения надежности приложений такого типа.

Web-сервис - (web-служба) – (англ. *web service*) – программная система, идентифицируемая строкой URI, чьи публичные интерфейсы и привязки определены и описаны языком XML. Описание этой программной системы может быть найдено другими программными системами, которые могут взаимодействовать с ней согласно этому описанию посредством сообщений, основанных на XML, и передаваемых с помощью интернет-протоколов. Сочетая различные web-сервисы, можно добиться достаточно большой гибкости применения, используя ограниченный набор web-сервисов. Кроме того, использование web-сервисов позволяет пользоваться услугами сервисов, расположенных на любой машине, к которой имеется доступ по HTTP-протоколу [1].

Увеличение сложности и объема разрабатываемых веб-систем приводит к повышению вероятности наличия в них невыявленных дефектов. Основным средством борьбы с дефектами программного обеспечения, в данном случае, систем сервис-ориентированной архитектуры, является верификация. Под верификацией программного обеспечения понимается процесс, направленный на подтверждения соответствия программного продукта заданным требованиям путем различного рода проверок и объективных доказательств [2]. Следует отметить, что процесс верификации должен носить комплексный характер, охватывающий все этапы разработки программного продукта (анализ требований, проектирование, кодирование, интеграцию), а также изменения, вносимые при сопровождении программного продукта в процессе его эксплуатации [3].

Основными методами верификации являются анализ программной документации (включая анализ программного кода) и тестирование программ. Под тестированием понимается процесс, помогающий определить корректность, полноту и качество разработанного программного обеспечения.

Особенности, связанные с технологией проектирования и разработки систем сервис-ориентированной архитектуры, обусловливают специфику тестирования и верификацию качества тестирования web-сервисов [4]. Анализ [4, 5], посвященных проблемам применения различных методов тестирования систем сервис-ориентированной архитектуры, показывает, что на сегодняшний день практически отсутствуют специализированные методы и инструментальные средства тестирования программных систем web-сервисов. Применяющиеся в процессе их разработки стандартные технологии тестирования зачастую не способны учесть особенности реализации таких сложных распределенных программных комплексов, каковыми являются системы сервис-ориентированной архитектуры.

Целью данной статьи является анализ соответствия методов тестирования и качества тестирова-

ния web-сервисов специфике разработки и применения систем сервис-ориентированной архитектуры.

Анализ методов тестирования и верификации качества тестирования web-сервисов

Анализ методов тестирования web-сервисов. Как было отмечено выше, тестирование наряду с анализом программной документации и исходного кода является основным методом верификации программного обеспечения. Кратко охарактеризуем применяющиеся методики тестирования, которые могут быть в той или иной степени применимы для систем web-сервисов.

Модульное тестирование – тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция.

Интеграционное тестирование – проверяет, есть ли какие-либо проблемы в интерфейсах и взаимодействии между интегрируемыми компонентами – например, не передается информация или же передается некорректная информация.

Системное тестирование – тестируется интегрированная система на её соответствие исходным требованиям. В свою очередь, системное тестирование включает следующие этапы:

1) *Альфа-тестирование* – имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком на стороне разработчика. Часто альфа-тестирование применяется для законченного продукта в качестве внутреннего приёмочного тестирования. Иногда альфа-тестирование выполняется под отладчиком или с использованием окружения, которое помогает быстро выявлять найденные ошибки. Обнаруженные ошибки могут быть переданы тестировщикам для дополнительного исследования в окружении, подобном тому, в котором будет использоваться ПО.

2) *Бета-тестирование* – в некоторых случаях выполняется распространение версии с ограничениями (по функциональности или времени работы) для некоторой группы лиц, с тем чтобы убедиться, что продукт содержит достаточно мало ошибок. Иногда бета-тестирование выполняется для того, чтобы получить обратную связь о продукте от его будущих пользователей.

При тестировании web-сервисов применяются также методы черного и белого ящиков (модульное тестирования, англ. Unit testing).

Метод черного ящика. Под методом черного ящика при тестировании web-сервисов подразумевается тестирование web-сервиса как единого целого, без учета внутренней структуры. Web-сервис должен быть протестирован как на стабильную работу в условиях подачи на вход валидных данных, на надежность и отказоустойчивость при подаче на вход некорректных данных. Тестовые наборы определяются в соответствии с функциональной спецификацией и постановкой задачи при написании данного web-сервиса, поэтому средства для автоматического тестирования методом черного ящика для web-сервисов практически отсутствуют.

Модульное тестирование подразумевает перенос части работы по тестированию на время кодирования. Во время кодирования разработчик создает специальные тесты (англ. Unit tests) индивидуально для каждого из разрабатываемых методов исходного кода. Данные тесты должны включать в себя все проверки, соответствующие специфике использования каждого из тестируемых методов.

Если «альфа-» и «бета-тестирование» относятся к стадиям до выпуска продукта (а также, неявно, к объему тестирующего сообщества и ограничениям на методы тестирования), тестирование «белого ящика» и «черного ящика» имеет отношение к способам, которыми тестировщик достигает цели.

Бета-тестирование в целом ограничено техникой черного ящика (хотя постоянная часть тестировщиков обычно продолжает тестирование белого ящика параллельно бета-тестированию). Таким образом, термин «бета-тестирование» может указывать на состояние программы (ближе к выпуску, чем «альфа»), или может указывать на некоторую группу тестировщиков и процесс, выполняемый этой группой.

Модульное тестирование web-сервисов. Как было отмечено выше, юнит-тестирование (модульное тестирование) – это процесс, позволяющий проверить на корректность отдельные модули исходного кода программы web-сервиса. Идея состоит в том, чтобы

писать тесты для каждой нетривиальной функции или метода. Это позволит достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, т.е. к появлению ошибок в уже написанных и оттестированных местах программы, а также облегчает локализацию и устранение таких ошибок.

Среди преимуществ юнит-тестирования следует упомянуть следующие:

1) Юнит-тестирование позволяет программистам проводить рефакторинг позже, будучи уверенными, что модуль по-прежнему работает корректно (регрессионное тестирование). Это поощряет программистов к изменениям кода, поскольку достаточно легко проверить, что код работает и после изменений.

2) Юнит-тестирование помогает устранить сомнения по поводу отдельных модулей и может быть использовано для подхода к тестированию «снизу вверх»: сначала тестируются отдельные части программы, затем программа в целом.

3) Юнит-тесты можно рассматривать как «живой документ» для тестируемого класса. Клиенты, которые не знают, как использовать данный класс, могут использовать юнит-тест в качестве примера.

4) Благодаря юнит-тестам возможно отделение интерфейса от реализации. Поскольку некоторые классы могут использовать другие классы, тестирование отдельного класса часто распространяется на связанные с ним другие классы. Это приводит к менее связанному коду и минимизирует зависимости в тестируемой системе.

Анализ методов верификации качества тестирования web-сервисов. Наиболее часто применяющимся на практике методом верификации качества тестирования web-сервисов является засев дефектов (инъекция ошибок, англ. fault injection) [4]. В ходе засева дефектов осуществляется внесение дефектов в исходный код готового программного продукта (web-сервиса) и проверка работоспособности его в условиях видоизмененного исходного кода. Засев дефектов является гибким методом, позволяющим выявить ошибки и дефекты, трудновыявимые для тестирования методами черного и белого ящиков. Засев дефектов также позволяет дать оценку качеству тестирования web-сервиса.

Анализ инструментальных средств модульного тестирования web-сервисов

Анализ средств модульного тестирования. Среды разработки web-приложений от наиболее известных производителей (Microsoft, Sun) предоставляют встроенные средства модульного тестирования (N-Unit и J-Unit соответственно) [6]. J-Unit – отрытая среда регрессионного тестирования, разработанная Э. Гаммой и К. Беком для проведения модульного тестирования web-приложений, написанных на языке Java. N-Unit — открытая среда юнит-тестирования приложений для .NET. Она была импортирована с языка Java (библиотека -JUnit). Первые версии N-Unit были написаны на J#, но затем весь код был переписан на C# с использованием таких новшеств .NET, как атрибуты. Данные средства предназначены для тестирования производительности модулей исходного кода приложения, но они ни в коей мере не соответствуют специфике написания и тестирования web-сервисов.

Существуют также известные расширения оригинального пакета N-Unit, большая часть из них также с открытым исходным кодом. N-Unit.Forms дополняет N-Unit средствами тестирования элементов пользовательского интерфейса Windows Forms. N-Unit.ASP выполняет ту же задачу для элементов интерфейса в ASP.NET. Данные инструментальные средства предназначены равно как для тестирования настольных приложений, так и web-приложений и web-сервисов. Различия по функциональности между двумя вышеназванными инструментальными средствами незначительны (состоят в основном в том, что J-Unit не подразумевает тестирование публичных – англ. public – методов).

Анализ инструментальных средств по инъекции дефектов. С учетом специфики тестирования web-сервисов разработаны инструментальные средства инъекции дефектов, а именно: Web Service – Fault Injection Technology (WS-FIT) [5] и System Under Benchmarking (SUB) and Benchmark Management System (SUB and BMS) [6].

Инструментальное средство WS-FIT. WS-FIT является инструментальным средством по инъекции ошибок на сетевом уровне в SOA приложениях, основанных на протоколе SOAP.

Данное инструментальное средство обладает широким спектром возможностей по инъекции ошибок, а также наблюдению результатов и построению отчетных графиков. В WS-FIT встроена система анализа полученных в результате тестирования данных, позволяющая дать объективную оценку степени отказоустойчивости тестируемого web-сервиса. WS-FIT позволяет выявить наиболее критичные к некорректным входным данным интерфейсы web-сервиса, что существенно ускоряет верификацию качества тестирования.

Инструментальное средство SUB and BMS. SUB and BMS подобно WS-FIT является средством автоматизации процесса верификации качества тестирования, позволяющим осуществлять инъекцию ошибок на сетевом уровне SOA с применением web-сервисов. Рассматриваемое инструментальное средство позволяет тестировать не только отказоустойчивость web-сервисов, но и всей системы, включая и HTTP-сервер. Аналитические возможности данного средства несколько меньше, чем у WS-FIT, но в отличие от предыдущего есть возможность интегрального тестирования отказоустойчивости системы в целом.

Заключение

В работе были рассмотрены существующие средства тестирования web-сервисов и стандартные средства их модульного тестирования. Необходимо отметить тот факт, что модульное тестирование при всей его гибкости и его наиболее полному соответствию идеологии построения систем сервис-ориентированной архитектуры нельзя рассматривать как универсальное средство тестирования web-сервисов. По определению, этот метод тестирует только модули сами по себе. Тем самым, ошибки интеграции, проблемы производительности или любые другие проблемы системы в целом не могут быть выявлены при использовании этого метода.

Кроме того, довольно трудно предугадать все варианты исходных данных, которые могут быть переданы модулю в реальной работе. Юнит-тестирование будет эффективным только при использовании совместно с другими способами тестирования. В ходе работы выявлено отсутствие специализированных инструментальных средств модульного тестирования web-сервисов, показаны недостатки имеющихся стандартных средств модульного тестирования.

Теоретические исследования в данной сфере следует проводить в плоскости исследования методов тестирования и технологических аспектов реализации web-сервисов, полное и качественное тестирование которых возможно соответствующими методами. Это позволит более точно очертить картину, складывающуюся при формировании тестовых заданий для верифицируемых систем сервис-ориентированной архитектуры, определить возможные «белые пятна» в построенной матрице и разработать технологию их полного перекрытия путем комбинирования различных методов тестирования.

Направлением для дальнейших исследований является детализация требований к инструментальным средствам для модульного тестирования web-сервисов и разработка специализированных средств модульного тестирования web-сервисов.

Литература

1. Kanter J. Understanding Thin-Client/Server Computing. – Redmond, WA: Microsoft Press, 1998. – 265 p.

2. Скляр В.В. Инструментальные средства для статистического анализа программного обеспечения: принципы применения, оценки и выбора // Электронное моделирование. – 2006. – Т. 28, № 2. – С. 29-41.

3. ДСТУ 3918-1999 [ИСО/МЭК 12207:1995]. Информационные технологии – Процессы жизненного цикла программного обеспечения. – К.: Госстандарт Украины, 2000. – 48 с.

4. MacDonald M., Szpuszta M. Pro ASP.Net 2.0 in C#2005 // ASPress. – 2005. – 380 p.

5. Looker N., Munro M. Practical Dependability Analysis of SOAP Based Systems // IST Publishing. – 2003. – №4. – P. 45-55.

6. Durães J., Vieira M., Madeira H. Dependability Benchmarking of Web-Servers // SAFECOMP. – 2004, LNCS 3219. – P. 297-310.

Поступила в редакцию 30.03.2007

Рецензент: д-р техн. наук, проф. В.С. Харченко, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.