УДК 004.054

В.П. Котляров, А.С. Иванов

МЕТОДИКА ТЕСТИРОВАНИЯ ВЫСОКОНАГРУЖЕННЫХ ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМ SMS ОПОВЕЩЕНИЯ

V.P. Kotlyarov, A.S. Ivanov

PRINCIPLES OF SMS INFORMING SYSTEM TESTING

Описана методика тестирования системы отправки SMS оповещений в роуминге, высылаемых в соответствии с требованиями Федерального агентства связи. Данная методика является адаптацией известных подходов к обеспечению качества программных продуктов для применения в высоконагруженных телекоммуникационных системах. Систематизированы результаты исследований, полученные в ходе разработки и запуска системы, обслуживающей 15 миллионов абонентов. Предложенная методика может применяться без существенных изменений для обеспечения качества систем информирования.

АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ; ТЕСТИРОВАНИЕ ПО МОДЕЛИ; ТЕСТИРОВАНИЕ В ВИРТУАЛЬНОМ ОКРУЖЕНИИ; АНАЛИЗ РАБОТЫ JVM; SMS ИНФОРМИРОВАНИЕ.

Paper describes principles of testing high load system of SMS informing in GSM roaming. Suggested technique is an adaptation of known approaches to quality assurance of software products for use in high-telecommunication systems. This paper summarizes the experience gained during the development and launch of the system serving 15 million subscribers. The principles can be applied to any other sms information system.

TEST AUTOMATION; MODEL-BASED TESTING; VIRTUAL MACHINE ENVIRONMENT; JVM PERFORMANCE; SMS INFORMING

В 2011 г. Федеральное агентство связи выпустило предписание всем операторам сети подвижной сотовой связи о необходимости информирования абонентов, выезжающих за пределы обслуживания домашней сети оператора. Существующая система информирования, использовавшаяся федеральными операторами связи, не позволяла информировать абонентов о стоимости услуг в роуминге, состоянии их лицевого счета и возможностях снижения затрат в роуминге. В связи с этим была поставлена задача разработать методику обеспечения качества создаваемого программного продукта, учитывающую объем обрабатываемых данных, возросшую сложность системы, взаимодействие системы с критически важными объектами инфраструктуры оператора (Home Location Register, SMS центр).

В рассматриваемой области были известны следующие подходы к обеспечению качества:

1. Функциональное тестирование [1]. Данный вид тестирования проверяет, что система выполняет функции, описанные в техническом задании (ТЗ) на систему. Стандартные подходы предусматривают построение плана тестирования и дальнейшее применение системы автоматизации тестирования или тестирование по модели. В крупных проектах оба подхода требуют значительных ресурсов. Более полным видится подход с использованием модели системы, позволяющий оперативно вносить в модель изменения под требования заказчика, а в некоторых методологиях – проводить верификацию модели. Использование тестирования по модели системы требует специального обучения персонала, что препятствует использованию данного подхода в компаниях, в которых обходятся без отдела контроля качества.

2. Тестирование производительности [1]. Данный вид тестирования используется для

проверки системы на соответствие требованиям к качеству сервиса. Это особенно важно в высоконагруженных распределенных системах, т. е. распределенных системах с ограничениями на время отклика и положительной динамикой нагрузки. Для нагрузочных тестов часто используются генераторы запросов на систему, у которых в процессе генерации необходимо учитывать, что характер нагрузки в реальных системах меняется в зависимости от времени суток и дня недели. Кроме того, часто генераторы нагрузки не учитывают зависимости между входными данными, что может изменить характер нагрузки.

3. Тестирование восстановления [1]. Данный тип тестирования проверяет, что система может сама восстанавливаться после аварий: проблем на сети, отключения внешних систем, повреждения конфигурационных и справочных данных.

4. Тестирование безопасности [1]. Проверяет соответствие системы стандартам и требованиям к безопасности. Обычно применяется, когда система имеет интерфейсы, взаимодействующие с публичными сетями. Этот вид тестирования в данной статье не рассматривается.

5. Тестирование совместимости [1]. Основная цель данного вида тестирования – проверка работы системы с конкретным видом оборудования. Часто проводится поставщиком оборудования и в настоящей работе не рассматривается.

6. Тестирование интерфейса и локализации [1]. Данный вид тестирования используется для тестирования создаваемых интерфейсов оператора и администратора системы, он здесь также не рассматривается.

Исследование решения поставленной задачи предполагало выполнение следующих шагов:

1) определить и обосновать методы обеспечения качества программного продукта для целей проекта;

2) адаптировать выбранные методы для тестирования высоконагруженных систем;

3) описать и автоматизировать методику тестирования;

4) выявить аспекты, плохо поддающиеся автоматическому тестированию и исследовать возможные варианты их преодоления.

Методика тестирования

Требования к качеству программного продукта были сформулированы в виде функциональных требований, требований к пропускной способности и требований к автоматическому восстановлению.

Проведены исследования подходов к тестированию каждой группы требований:

1. Согласно ТЗ система функционирует следующим способом: при изменении местоположения абонента в роуминге система запрашивает по абоненту дополнительную информацию из внешних систем и по заданным критериям определяет шаблон SMS сообщения. Данный шаблон может содержать специальные значения – маркеры, подставляемые системой, исходя из информации о стране пребывания, стоимости услуг, оператора регистрации.

Тестирование системы с полным перебором возможных состояний для данной системы было нереальным из-за сложной логики выбора шаблонов сообщений и различных вариантов подстановки специальных маркеров в шаблоны. Все это приводило к экспоненциальному росту числа возможных последовательностей событий, которые должны были бы тестироваться. Для проведения функционального тестирования системы было выбрано эффективное решение в виде двух моделей: первая – для тестирования возможности выбора всех шаблонов с фиксированными маркерами подстановки, вторая – для генерации всех возможных подстановок маркеров в фиксированный шаблон.

В изначальном плане тестирования не существовало требований к максимальному количеству сообщений на абонента. В результате проведенного исследования такое требование в план было введено.Тестирование проводилось после тестирования стабильности, где в результате недельного прогона фиксировались отправляемые сообщения, которые в дальнейшем проверялись на соответствие входным данным, на ограничение количества отправляемых сообщений.

2. Для оценки пропускной способности системы были применены следующие виды тестирования:

нагрузочное тестирование, обеспечивающее оценку времени обработки данных;

стресс-тестирование или тестирование стабильности, обеспечивающее оценку запаса производительности системы;

конфигурационное тестирование, обеспечивающее оценку влияния окружения на работу системы.

Анализ системы в ходе тестирование проводился на основе сбора системных метрик, сбора данных от среды исполнения (Java Virtual Machine), анализа времени ответа и пропускной способности всей системы. Предполагалось, что анализ системы может внести дополнительные издержки и исказить данные тестирования, по этой причине было запланировано повторное тестирование без анализа системных метрик и метрик JVM со сравнением результатов по критерию Стьюдента.

Нагрузочное тестирование используется для оценки времени обработки входных данных. При данном виде тестирования эмулируется обычный входной поток данных. Для времени обработки строится функция распределения, рассчитываются квантили. В качестве оценочного значения выбран квантиль при 99 % вероятности. Система написана на языке Java, для которого свойственны задержки при запуске сборки мусора. Чтобы оценить влияние данных пауз, производился анализ работы сборщика мусора, анализ swap системы [4].

Стресс-тестирование применяется для оценки запаса прочности системы относительно ожидаемой функциональности. Данный вид тестирования также предполагает сбор широкого спектра системных метрик, анализ которых позволяет выявить максимальную нагрузку на систему, сформулировать критические уровни нагрузки для системы мониторинга и выявить узкие места системы.

Тестирование стабильности предполагает работу под обычной нагрузкой на протяжении длительного периода времени. В нашем случае на суточном и недельном интервалах времени. Данный вид тестирования позволяет выявить влияние редких событий, например, запуск систем архивации или задач Cron, на функционирование системы, а также выявить утечки системных ресурсов и проблемы Class Loader в Java.

Конфигурационное тестирование. В задаче тестирования пропускной способности системы происходил анализ влияния выделяемых ресурсов виртуальной машины на общую пропускную способность. Данный вид тестирования сводился к стресс-тестированию при изменяемых параметрах виртуальной машины.

3. Тестирование автоматического восстановления. Необходимо отметить, что тестируемая система должна функционировать на виртуальной машине, по этой причине анализ восстановления после аппаратных сбоев и миграции виртуальных машин в данной статье не рассматривается. В рамках тестирования автоматического восстановления системы протестировано восстановление отдельных процессов после экстренного завершения системы (kill -9 <pid>), в т. ч. автоматический запуск базы данных и ее диагностика. Восстановление доступа к внешним системам после сбоев в сети. Протестирован автоматический запуск системы после нормальной и экстренной перезагрузки системы.

Адаптация к нагрузке

Предполагалось, что нагрузочное тестирование системы удастся провести, используя реальный сетевой трафик в качестве входных данных, однако, выполнить зеркальное отображение трафика в сеть, где находилась тестовая лаборатория, не удалось. Также не удалось перенести недельный трафик из-за отсутствия требуемых ресурсов для хранения данных. В результате исследований было решено разбить тестирование на два этапа: тестирование с генерацией трафика в тестовой лаборатории и тестирование с «зеркалированием» трафика на этапе опытной эксплуатации.

Для создания генератора трафика был проведен анализ абонентов, выезжающих в роуминг (рассматривались такие параметры, как частота выезда; продолжительность пребывания; сообщения, посылаемые

в сигнальной сети оператора; количество посещаемых стран абонентом; характерные маршруты; аномальное поведение) [3]. В результате было создано несколько моделей в виде машин состояний, описывающих поведения абонентов в роуминге. В начале каждого 5-минутного интервала времени тестовой системой создавались новые тестовые абоненты в роуминге, дополнительно выбиралась часть абонентов, уже хранящихся в системе, по всем выбранным абонентам происходил переход в рамках их машины состояний и создавалось соответствующее событие, которое посылалось в системе. Данный подход позволил хранить только необходимый минимум информации (id абонента, id его машины состояний, id состояния в рамках машины состояний), а также позволил воссоздать воздействия, которые посылались на систему. Формально показано, что использование предложенного генератора трафика создает однотипную нагрузку с реальным потоком данных [5].

Автоматизация

Методика тестирования изначально разрабатывалась с учетом необходимости автоматизации рутинных, повторяющихся задач.

Цикл автоматизированного тестирования состоял из следующих этапов.

1. Запуск виртуальной машины из готового образа.
2. Назначение роли (типа тестирования) на виртуальной машине.
3. Настройка общего окружения средствами системы управления конфигурациями.
4. Настройка окружения в зависимости от роли (например, для анализа работы Java Virtual Machine на системе запускались сервисы jstatd, jfr).
5. Загрузка и сборка тестируемой системы из системы контроля версий.
6. Загрузка и запуск тестов в соответствии с ролью тестируемой системы.

На ручную работу оставалось только-создание модели тестирования, настройка окружения и анализ результатов тестирования. Дополнительно, в ручном режиме просматривались журналы недельных тестов на предмет странного/подозрительного поведения, т. е. поведения системы, которое является ошибкой, но не описано ни в ТЗ, ни в тестовом плане (например, множественные отправки SMS одному человеку на пограничных областях.

Проблемы тестирования

Необходимо отметить, что данный подход не лишен недостатков, которые, к счастью, не влияют на основные результаты тестирования. В частности, не отражено тестирование систем мониторинга – важной части промышленной системы, что не было предусмотрено требованиями сдачи системы в эксплуатацию. Часть ошибок была найдена только в процессе ручного анализа журналов тестирования. Поскольку понятие странное/аномальное поведение не формализовано, то идентификация связанных с ним ошибок возможна только человеком, который по результатам их анализа создает новые требования в программе тестирования. Использование поведенческих моделей позволяет гарантировать тестирование программного продукта по критерию путей и, в случае необходимости, провести верификацию свойств модели [2].

Результаты

Исходный код, исследуемый в рамках методики, имеет объем 37 KLOC.

Выше рассматривались причины, по которым функциональное тестирование было разбито на два блока. Рассмотрим метрики тестирования по каждому направлению (табл.).

С помощью нагрузочного и стресс-тестирования получены следующие результаты:

- пропускная способность системы (предполагаемая нагрузка) 8 000–10 000 TPS;
- максимальная пропускная способность 18 000 TPS;
- среднее время обработки запроса 290 мс;
- дисперсия времени обработки 30 мс.

В ходе конфигурационного тестирования был представлен план выделения ресурсов виртуальной машине для задач масштабирования.

Метрики тестирования

Тестируемый функционал	Количество обнаруженных ошибок	Покрытие по путям, %
Тестирование шаблонов	108	100
Тестирование маркеров	24	100

В процессе тестирования восстановления ошибок не обнаружено.

Предложенная методика тестирования позволила с минимальными замечаниями начать этап опытной эксплуатации телекоммуникационной системы, обслуживающей 15 млн абонентов.

В дальнейшем предлагаемая методика может быть расширена использованием UCM поведенческих моделей для генерации трас при функциональном тестировании. Данная методика может быть применена и в области нотификации абонентов: оповещения МЧС, сообщения городской администрации и оповещения о стоимости услуг.

СПИСОК ЛИТЕРАТУРЫ

1. **Майерс Г., Баджетт Т., Сандлер К.** Искусство тестирования программ. 3-е изд. М.: Диалектика, 2012. № 3(174). 272 с.

2. **Никифоров И.В., Дробинцев П.Д., Котляров В.П.** Методика проектирования тестов сложных программных комплексов на основе структурированных UCM моделей // Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. СПб.: Изд-во СПбГПУ, 2013. № 3(174). С. 99–105.

3. **Janert Ph.K.** Data Analysis with Open Source Tools. Sebastopol: O'Reilly Media Inc., 2011.

4. **Oaks Sc., Blanchette M.** (eds.). Java Performance: The Definitive Guide. Sebastopol: O'Reilly Media Inc., 2014.

5. **Stewart W.J.** Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modelling. Woodstock: Princeton Univercity Press, 2009.

REFERENCES

1. **Mayers G., Badzhett T., Sandler K.** *Iskusstvo testirovaniya programm*, Moscow: Dialektika Publ., 2012, No. 3(174), 272 p. (rus)

2. **Nikiforov I.V., Drobintsev P.D., Kotlyarov V.P.** Metodika proyektirovaniya testov slozhnykh programmnykh kompleksov na osnove strukturirovannykh UCM modeley [Formal models structurization based technique of complex software projects testing], *Nauchno-tekhnicheskiye vedomosti SPbGPU. Informatika. Telekommunikacii. Upravlenie.* St. Petersburg: SPbGPU Publ., 2013, No. 3(174), Pp. 99–105. (rus)

3. **Janert Ph.K.** *Data Analysis with Open Source Tools.* Sebastopol: O'Reilly Media Inc., 2011.

4. **Oaks Sc., Blanchette M.** (eds.). *Java Performance: The Definitive Guide.* Sebastopol: O'Reilly Media Inc., 2014.

5. **Stewart W.J.** *Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modelling.* Woodstock: Princeton Univercity Press, 2009.

КОТЛЯРОВ Всеволод Павлович – *профессор кафедры информационных и управляющих систем Санкт-Петербургского государственного политехнического университета, кандидат технических наук.*
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29.
E-mail: vpk@spbstu.ru

KOTLYAROV, Vsevolod P. *St. Petersburg State Polytechnical University.*
195251, Politekhnicheskaya Str. 29, St. Petersburg, Russia.
E-mail: vpk@spbstu.ru

ИВАНОВ Александр Сергеевич – *аспирант кафедры информационных и управляющих систем Санкт-Петербургского государственного политехнического университета.*
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29.
E-mail: al.s.ivanov@gmail.com

IVANOV, Aleksandr S. *St. Petersburg State Polytechnical University*.
195251, Politekhnicheskaya Str. 29, St. Petersburg, Russia.
E-mail: al.s.ivanov@gmail.com