Тестирование протоколов электронной почты Интернета с использованием моделей

Н. В. Пакулин, А. Н. Тугаенко
{npak, tugaenko}@ispras.ru

Аннотация. В статье рассматриваются вопросы тестирования почтовых протоколов с использованием формальных моделей протоколов: предложен метод моделирования почтовых протоколов, рассмотрены особенности почтовых протоколов в контексте тестирования, представлены результаты тестирования популярных почтовых серверов с открытым кодом. В качестве примера представлена разработка тестовых наборов для протоколов SMTP и POP3 на языке JavaTESK – расширении языка Java, реализующем тестирование с формальными методами. Тестовые наборы состоят из двух частей: независимого тестирования соответствия протоколов спецификациям и совместного теста, имитирующего работу почтовых протоколов в сети.

Ключевые слова. Формальные методы, верификация, тестирование, электронная почта, UniTESK.

1. Введение

Электронные письма являются основой современного взаимодействия между людьми. Миллионы писем перемещаются ежедневно в сети Интернет. Надежность и корректность инфраструктуры почтовых сообщений чрезвычайно важна для современного информационного общества. В этой статье мы коснемся двух аспектов этих вопросов: надежности (1) передачи почты в сети Интернет и (2) доставки писем конечным адресатам.

На своем пути от отправителя к получателю почтовое сообщение проходит через цепочку промежуточных серверов. Часто промежуточные сервера поставляются разными разработчиками, и поэтому они имеют различные реализации почтовых протоколов. Общая надежность механизма передачи электронной почты в значительной степени определяется совместимостью различных реализаций серверов между собой, а также корректностью функциональной части каждой реализации.

В настоящее время тестирование соответствия реализации стандартам служит основным методом обеспечения их совместимости. Логическое обоснование такого предложения основано на предположении хорошего проектирования протокола: если две реализации соответствуют спецификации протокола, то

они совместимы. В этой статье мы не обсуждаем, выполняется ли это предположение для почтовых протоколов сети Интернет, мы предлагаем подход для тестирования соответствия таких протоколов.

Несмотря на более чем двадцатилетнюю историю почтовых протоколов и существования десятков реализаций протоколов SMTP[1], POP3[2] и IMAP4[3], до сих пор нет открытого и независимого от реализации тестового набора для проверки их соответствия стандартам. Мы считаем, что есть несколько причин объясняющих это:

1. почтовые протоколы выглядят простыми;
2. разработчики при тестировании фокусируются на аспектах разработки почтовых серверов, не имеющих отношения к соответствию стандартам: настройка параметров, безопасность, внутреннее хранилище сообщений, быстродействие и так далее.

Рассмотрим эти причины более детально. Во-первых, простота почтовых протоколов кажущаяся. Можно легко увидеть, что протоколы SMTP, POP3 и IMAP4 содержат ряд нетривиальных особенностей:

1. почтовые протоколы недоспецифицированы, существенная часть функциональности оставлена на усмотрение разработчика, в спецификации описаны несколько вариантов возможного дальнейшего поведения системы;
2. почтовые протоколы недетерминированы: стандарт допускает различные варианты поведения системы, включая отказ в доставке почтовых сообщений или разрыв соединений;
3. функции почтовых протоколов различаются по степени обязательности (MUST, SHOULD, MAY и прочие);
4. архитектура протоколов расширяемая: реализация протокола может использовать различные расширения, как дополняющие функциональность протокола, так и изменяющие её.

Перечисленные особенности определяют фактическую сложность разработки тестовых наборов для тестирования реализаций протоколов на соответствие спецификациям.

Во-вторых, дополнительная сложность в разработке почтовых серверов связана с поддержкой большого числа настроек, необходимых для практической эксплуатации – параметров пересылки, аутентификации, безопасности, хранилища писем и так далее. Мы изучили тестовые наборы нескольких реализаций, построенных по модели открытого кода: Apache James, Sendmail, Dovecot, Qmail, Postfix. Как оказалось, тесты сильно связаны с конкретными реализациями и непереносимы для тестирования серверов от других разработчиков, так как используют особенности реализации: настройки параметров, доступ к внутреннему состоянию, выполнение в одном

процессе вместе с реализацией. Использование тестов, разработанных для одной реализации, не предоставляется возможным для проверки других реализаций. Более того, как показал анализ, эти тесты не предназначены для тестирования соответствия, они проверяют, прежде всего, корректность реализации многочисленных настроек почтового сервера, не относящихся напрямую к стандартам SMTP, POP3 и IMAP4. Проблема заключается в том, что такой подход к тестированию не гарантирует обеспечения совместимости сервера со стандартом. В частности, именно тестирование соответствия позволило выявить в почтовом сервере James серьезную функциональную ошибку – зацикливание передачи сообщений в определенной конфигурации.

Строгая связь между тестами и требованиями стандартов позволяет оценить полноту тестирования с точки зрения внешнего пользователя почтового сервиса. Как показывает пример с сервером James, тщательное тестирование внутренних функций реализации не гарантирует качества функционирования реализации в реальном окружении.

Мы считаем, что задача тестирования соответствия стандартам почтовых протоколов обладает существенным практическим значением. Целью нашего исследования является разработка новой архитектуры тестовой системы, а также методики для разработки элементов системы, пригодных для учета особенностей почтовых протоколов. В данной работе представлена новая технология разработки тестов для тестирования соответствия почтовых протоколов. Данная технология включает несколько этапов разработки тестов, причем этапы сконструированы таким образом, что каждый этап может быть завершающим. Новизна метода тестирования соответствия почтовых протоколов заключается в последовательной трансформации модели тестирования, при которой на каждом шаге получаются отлаженные тесты (модель и программа зависят от шага). В начале разработки тестового набора пишется набор элементарных испытаний (test cases) или берется уже имеющийся. На следующих шагах строится интерфейсная модель, в построенную модель добавляются состояния. После выполнения этих шагов получается тот же по функциональности тестовый набор, что был после первого шага. Но на данном этапе уже намного проще расширять тестовый набор путем расширения формальной спецификации тестируемого протокола. Предложенный подход, основанный на методологии тестирования на основе моделей, фокусируется только на тестировании соответствия и не рассматривает другие аспекты проверки почтовой инфраструктуры, такие как тестирование взаимодействия, производительности, надежности и прочие.

В качестве примера применения предложенного метода с его помощью был разработан открытый тестовый набор для проверки соответствия основной функциональности протоколов SMTP, POP3 и IMAP4 их стандартам. Также был разработан тест, использующий композицию спецификаций протоколов SMTP и POP3. Данный тест используется для проверки корректности взаимодействия реализаций различных почтовых серверов.

Предложенный подход позволяет последовательно трансформировать тесты, написанные как элементарные испытания, в тесты, сгенерированные на основе моделей, с сохранением функциональности и работоспособности (отлаженности) тестов. На первых шагах происходит только трансформация подхода тестирования, формируется модель тестируемого протокола, в то время как последующие шаги метода позволяют легко расширять набор тестов для проверки определенной функциональности.

Структура статьи: в разделе 2 представлен краткий обзор современных почтовых протоколов, в разделе 3 кратко рассмотрены существующие подходы к тестированию протоколов и обсуждается выбор тестирования на основе моделей и технологии автоматизированного тестирования UniTESK для предлагаемого подхода. Раздел 4 содержит краткое введение в технологию UniTESK, которая лежит в основе предлагаемого подхода. В разделе 5 описан подход, использующийся для разработки тестовых наборов для SMTP, POP3 и IMAP4. В разделе 6 представлены разработанные к настоящему моменту тестовые наборы для протоколов SMTP, POP3 и IMAP4. В разделе 7 обсуждаются плюсы и минусы предложенного подхода. Раздел 8 содержит полученные к данному моменту результаты и описывает направления будущих исследований.

2. Краткое введение в почтовые протоколы

Большинство почтовых сообщений в сети Интернет передаются посредством протокола SMTP [1] – сетевого протокола верхнего уровня стека протоколов TCP/IP. Это текстовый протокол, состоящий из двух частей: клиента и сервера. Клиент подает команды и сервер исполняет их, возвращает код ответа и другие данные, если это необходимо. Протокол SMTP имеет свою подсеть в стеке TCP/IP, содержащую множество почтовых серверов и агентов пересылки, используемых для передачи сообщений между различными сетевыми узлами. Особенность SMTP состоит в том, что каждый физический сервер может функционировать и как SMTP клиент, и как SMTP сервер: будучи сервером он принимает входящее сообщение и становится клиентом для пересылки сообщения следующему узлу.

Протокол SMTP используется для отправки сообщений, но когда реализация SMTP сервера идентифицирует себя как конечного получателя почтового сообщения, пересылка прекращается, а письмо помещается во внутреннее, зависящее от реализации, хранилище. Для получения сообщений из хранилища конечные пользователи используют другие протоколы: POP3 [2] и IMAP4 [3]. Оба протокола также являются текстовыми, поддерживающими роли клиента и сервера. Клиенты получают доступ к хранилищу, подавая протокольные команды серверам, и сервера предоставляют необходимую информацию в ответах на эти команды. Обычно реализации POP3 и IMAP4 поддерживают только одну роль в одно время – либо клиента, либо сервера.

Также стоит отметить, что реализации почтовых серверов могут функционировать на одном сетевом узле или даже в одном почтовом сервере. В частности, почтовые сообщения, пересылаемые по протоколу SMTP, доступны клиентам по протоколам POP3 и IMAP4. По этой причине при тестировании необходимо рассматривать почтовый сервер как совокупность реализаций нескольких протоколов, причем тестовые воздействия возможны сразу по нескольким протоколам.

3. Тестирование почтовых протоколов

В современной индустрии тестирование реализаций протоколов на соответствие стандартам большей частью основано на ручной разработке тестовых наборов, состоящих из отдельных программ, написанных на специализированных или обычных языках программирования. Такие программы называются элементарными испытаниями (test case), в них реализуется генерация последовательности стимулов, подача сгенерированной последовательности в целевую систему, считывание и анализ реакций [4].

Создателями реализаций почтовых серверов разработали многочисленные тестовые наборы для функционального тестирования своих реализаций. Эти тестовые наборы написаны, как правило, на том же языке, который использовался для создания сервера (C, C++, Java и т.д.). Они нацелены прежде всего на тестирование внутренних механизмов реализаций – обработки всевозможных опций настройки сервера, обработки ошибок, взаимодействия с пользователем и т.п. Тесты, созданные разработчиками серверов, не содержат четко определенного подмножества тестов соответствия спецификациям; кроме того, тесты не переносимы между реализациями почтовых приложений, так как существенно используют знание о внутреннем устройстве конкретной реализации.

Стоит отметить разработанный в Apache Project инструмент Mail Protocol Tester (MPT) [5], предназначенный для тестирования произвольных протоколов с тестовыми сообщениями. Тесты задаются на языке XML; спецификация тестов не зависит от какой-то конкретной реализации. Тест представляет собой последовательность строковых команд и описаний ожидаемых ответов реализаций. Ожидаемые ответы описываются регулярным выражением — если ответ сервера не соответствует регулярному выражению, MPT останавливает работу и выдает сообщение об ошибке. MPT не поддерживает ветвления в тестах, циклы и использование параметров.

Исследователи обычно рассматривают элементарные испытания как один из традиционных методов тестирования [6-8], в то время как тестирование на основе моделей рассматривается как новый метод, решающий множество неразрешаемых с помощью традиционных методов проблем (например, неточность в вычислении покрытия функциональности тестируемой системы или ручная разработка прослеживаемости требований). Предложенный в этой статье метод рассматривает тестирование на основе моделей как расширение

метода, основанного на элементарных испытаниях. В качестве демонстрации предложенного метода представлены разработанные тестовые наборы для протоколов SMTP, POP3 и IMAP4. Также проводилась разработка тестового набора для проверки корректности взаимодействия реализаций почтовых протоколов SMTP и POP3.

Рассмотрим требования к тестовому набору. Тестовый набор для тестирования соответствия должен обладать следующими свойствами:

1. прослеживаемостью требований. Тесты (полученные из формальной спецификации) должны соотноситься с требованиями стандарта, должно быть наглядно видно, какие требования какими тестами покрываются.

2. многообразием настроек на особенности реализаций (MAY, SHOULD, MUST и другие). Должна быть опция для определения множества требований, поддерживаемых тестируемой реализацией. В это множество не должны попадать требования, не поддерживаемые тестируемой реализацией.

3. полнотой тестового набора в смысле покрытия требований. Полученный в итоге тестовый набор должен покрывать как минимум все выделенные из стандарта требования, помеченные в стандарте как обязательные для каждой реализации.

Тестирование на основе элементарных испытаний не предоставляет возможностей явной прослеживаемости требований, полнота тестирования в смысле покрытия всех требований при таком подходе также затруднена. Походы, основанные на TTCN3 [9] и JUnit [10], требуют внешних инструментов для установления соединения между тестами и требованиями, таких как матрицы прослеживаемости (traceability matrixes). Слабость таких инструментов в том, что требования не являются составной частью теста.

Кроме того, разработка большого числа тестов приводит к дублированию кода или сложным и запутанным связям. Нужно использовать методы, позволяющие декомпозировать тестовый набор и обеспечить прослеживаемость требований.

Необходимые возможности дают инструменты, построенные на формальных методах. Выделение формальных спецификаций позволяет:

1. задавать формальные связи между требованиями и тестами, автоматически отслеживать качество тестирования в терминах покрытия спецификации;

2. использование формальной спецификации дает возможность многократно использовать модель для проверки правильности поведения реализаций;

3. наличие спецификации позволяет генерировать тестовые воздействия в терминах модели и автоматически фильтровать избыточные воздействия.

Также при выборе метода для построения тестовых последовательностей важно учитывать особенности почтовых протоколов. В частности, из-за недетерминированного поведения протоколов и из-за недоспецифицированности протоколов для тестирования следует выбирать подходы, обеспечивающие построение тестовых последовательностей с учетом откликов целевой системы. Автоматическое вынесение вердикта из постусловий спецификации может решить проблему проверки корректности поведения целевой системы.

Существует много инструментов и подходов для тестирования. NModel [11] описывает модельную систему на языке C# и предоставляет основные возможности автоматической генерации тестов («на лету», on-the-fly testing) и максимизации покрытия согласно специально написанному критерию. Но для использования автоматической генерации тестов необходимо написать отдельную программу, описывающую сложный алгоритм обхода автомата теста.

Предыдущая версия SpecExplorer (2004) [12] предоставляла возможность тестирования «на лету», но эта версия больше не поддерживается. В последней версии SpecExplorer(2010) [13] тестирование «на лету» не документировано. Инструмент Conformiq Qtronic [14] не поддерживает тестирование «на лету», вместо этого он генерирует тестовые сценарии TTCN-3.

Технология UniTESK [15,16] поддерживает нотацию формальных спецификаций, автоматическую генерацию («на лету») тестовых воздействий (на уровне кода, нет необходимости писать отдельную программу) и автоматический анализ результатов. Поэтому мы решили использовать эту технологию в нашем проекте. UniTESK предоставляет средства для формальной спецификации асинхронных систем [17] в виде контрактных спецификаций переходов расширенного конечного автомата [18-20]. Помимо программных систем средствами UniTESK возможно специфицировать и тестировать аппаратные системы, такие как микропроцессоры [21,22].

В технологии UniTESK формальные спецификации, формализующие требования в виде пред- и постусловий, используются для генерации тестовой последовательности [23]. Также для генерации тестовых последовательностей должен быть определен конечный автомат (автомат теста).

Процесс тестирования в UniTESK представляет собой автоматический обход автомата теста, при котором наблюдаемое поведение реализации автоматически верифицируется тестовыми оракулами, сгенерированными из формальной спецификации. Использование формальных спецификаций позволяет автоматизировать проверку корректности поведения реализации и оценку полноты тестирования, а представление теста как автомата дает

возможность автоматически генерировать длинные и разнообразные последовательности тестовых событий.

В связи с тем, что реализации различных почтовых протоколов часто функционируют на одном и том же сетевом узле или в одном почтовом сервере, в дополнение к тестированию соответствия имеет смысл проводить интеграционное тестирование, направленное на проверку корректности взаимодействия реализаций различных протоколов, в частности, на корректность обработки данных. Инструмент JavaTESK [24] позволяет при наличии уже разработанной программы для генерации тестовых наборов для тестирования соответствия почтовых протоколов без особых дополнительных усилий создавать тесты для проведения интеграционного тестирования. При таком подходе используется более половины кода, написанного для проведения тестирования соответствия.

Авторы использовали представленный метод для разработки тестовых наборов для тестирования соответствия реализаций протоколов SMTP, POP3 и IMAP4, а затем и интеграционного тестирования реализаций различных почтовых протоколов, функционирующих на одном компьютере. Из инструментов, реализующих подход UniTESK, был выбран JavaTESK. JavaTESK использует язык программирования Java с набором расширений для записи формальных спецификаций и задания тестов.

4. Обзор технологии UniTESK

Стандартным форматом нормативной документации протоколов Интернет являются документы RFC (Request for Comment). Требования в этих документах изложены на английском языке и представляют собой неформальный текст, описывающий желаемое поведение системы. В рамках технологии UniTESK для формальной записи требований используются спецификационные расширения языков программирования – Java или C.

Запись неформальных требований нормативной документации на формальном языке представляет собой модель протокола. В подходе UniTESK формальная модель протокола строится в терминах конечных автоматов. Переходы между состояниями могут задаваться как в явном виде, так и в неявном. В случае явного задания перехода модель содержит алгоритм вычисления следующего состояния и реакции протокола; неявное задание перехода представляет собой предикат, который накладывает ограничения на допустимые конечные состояния и реакции протокола.

Спецификация на языке JavaTESK обычно состоит из одного или нескольких спецификационных классов, которые описывают состояния и переходы моделируемого протокола. Переходы протокола представляются как методы класса специального вида (спецификационные методы), кроме того поддерживается возможность задать ограничения на множество допустимых

состояний посредством инвариантов типов (ограничений на значения типов) и инвариантов переменных состояния.

Неявное задание переходов реализуется в виде пред- и постусловий. В предусловии содержатся ограничения на допустимые значения параметров воздействия на целевую систему и на состояние, в котором воздействие оказывается. На воздействия целевая система может реагировать изменением состояния, подачей реакции или и тем, и другим. Постусловие проверяет допустимость продемонстрированного поведения системы.

При моделировании поведения тестируемой системы используется набор структур данных, которые называются абстрактными состояниями. Для вынесения вердикта о корректности поведения целевой системы UniTESK использует данные, которые содержатся в абстрактном состоянии модели.

В UniTESK воздействия на целевую систему и реакции целевой системы описываются в терминах модели, содержащейся в спецификациях. Поэтому для взаимодействия модельной и целевой систем используется некоторый посредник – медиатор, – который переводит параметры воздействия из модельного представления в представление реализации, реакции целевой системы в модельное представление, а также при необходимости вносит изменения состояния целевой системы в абстрактное состояние.

Тестовый сценарий определяет последовательность воздействий, оказываемых на целевую систему. В качестве теоретической основы для построения сценариев используется метамодель конечных автоматов. В JavaTESK автомат теста задается в сценарном классе, который содержит процедуру определения текущего состояния автомата теста и итераторы тестовых воздействий. Инструмент JavaTESK предоставляет набор обходчиков, которые строят цепочки тестовых воздействий из описания автомата теста.

5. Предложенный метод тестирования почтовых протоколов

Формально процесс тестирования почтовых протоколов можно разбить на две части: независимое тестирование реализаций на соответствие стандартам (тестирование соответствия) и интеграционное тестирование реализаций различных протоколов. Первая часть заключается в тестировании соответствия различных реализаций серверов каждого протокола требованиям, независимо от реализации данным сервером других протоколов. Здесь внимание уделяется корректности функциональных частей протоколов. Вторая часть представляет собой интеграционное тестирование реализаций различных почтовых протоколов. В интеграционном тестировании внимание уделяется взаимодействию реализаций разных протоколов, в частности, корректности обработки данных.

Почтовые протоколы могут находиться в нескольких состояниях. При получении определенных стимулов они генерируют и передают отклики, а

также переходят в другие состояния либо остаются в тех же. С учетом этого на базе тестирования UniTESK был разработан метод для тестирования соответствия почтовых протоколов. Данный метод основан на методе тестирования, изложенном в работе [25], но в отличие от разработанного там метода позволяет разрабатывать тестовые наборы путем последовательного перехода от тестирования на основе элементарных испытаний (test cases) к тестированию на основе моделей, если уже имеется набор элементарных испытаний, или как метод последовательного создания тестов на основе моделей. Разработанный метод содержит следующие основные шаги:

1. знакомство с предметной областью, разработка примеров и элементарных тестов. Данный шаг не дает никаких видимых результатов, но он важен для более детального понимания работы протоколов и помогает при реализации последующих шагов метода.

2. составление каталога требований. Каталог требований представляет собой базу данных или таблицу с описанием требований. При выделении требования из RFC в таблицу или базу данных вместе с текстом требования заносятся идентификатор требования, а также тип требования (синтаксическое или функциональное), обязательность выполнения данного требования (в RFC часть требований помечена ключевыми словами MUST, SHOULD, MAY и другими, характеризующими обязательность выполнения требования реализациями), ссылка на место в RFC и, возможно, какие-то дополнительные атрибуты.

3. построение упрощенной модели протокола, создание экспериментального теста – автомата теста с единственным состоянием. Упрощенная модель протокола включает в себя знания о командах протокола, а также о возможных реакциях на эти команды. Экспериментальный тест состоит из спецификационного, медиаторного и сценарного классов. Спецификационный класс на данном этапе содержит только сигнатуры методов, а все проверки производятся в сценарном классе. Такой тест называется линейным, подача стимулов и считывание реакций в нем производятся в определенном порядке, указанном в сценарии.

4. построение концептуальной модели поведения протокола; выделение основных состояний протокола; построение автомата теста с выделенными состояниями; добавление в экспериментальный тест блока, ответственного за перевод модельной системы между состояниями. Концептуальная модель описывает внешне наблюдаемое поведение системы как операции над некоторым набором абстрактных компонентов и объектов, составляющих её. Эти компоненты используются только для моделирования поведения и могут не соответствовать разбиению самой системы на компоненты.

Добавление в тест блоков, переводящих систему из одного состояния в другое, преобразует тест из линейного в автоматный. В таком тесте построение цепочки тестовых воздействий происходит при обходе автомата, а подача стимулов и считывание реакций производятся только из разрешенных состояний.

5. формализация требований, перенос проверки полученных от реализации откликов в отдельный блок – спецификационный класс. При формализации требований проверяется их полнота и непротиворечивость. Итогом шага является формальная спецификация протокола, написанная на расширении одного из языков программирования.

6. расширение сценария и спецификации для покрытия всех требований. В сценарном классе описываются стимулы, которые будут подаваться системе. Порядок подачи стимулов формируется при обходе автомата и зависит от условий подачи стимулов в определенных состояниях. Обычно один сценарный класс отвечает за какой-либо конкретный раздел требований. Для покрытия всех формальных требований может потребоваться несколько сценарных классов.

7. выполнение тестов и анализ полученных результатов. Анализ результатов может показать, что не все требования покрываются сгенерированным тестовым набором. Если это так, то шаг 6 повторяется до тех пор, пока все требования не будут покрыты

Если на момент начала написания тестов уже имеется набор элементарных испытаний (test cases), а также если тестирование проводит человек, уже имеющий опыт создания тестов на основе моделей, то первый шаг можно опустить. Также стоит отметить, что данный порядок шагов не является единственным. Так, шаги 3 и 4 могут быть объединены, а шаг 5 может выполняться параллельно с шагами 3 и 4. Также возможны итерации в рамках подцепочек приведенной последовательности.

Интеграционное тестирование проводится после того, как готовы спецификационные и медиаторные классы для тестирования соответствия протоколов, участвующих в интеграционном тестировании. При разработке тестовых наборов для тестирования интеграции в качестве спецификации используется композиция спецификаций тестируемых протоколов, а медиаторные классы остаются без изменений. Сценарный класс описывает определенную последовательность действий, которая позволяет проверить корректность передачи, получения и обработки данных различными протоколами.

6. Применение метода для тестирования протоколов SMTP, POP3 и IMAP4

В данном разделе описывается применение метода для тестирования почтовых протоколов. Сначала описывается разработка тестового набора для тестирования соответствия почтовых протоколов, затем – разработка тестов для проведения интеграционного тестирования.

6.1. Независимое тестирование реализаций протоколов

Первым шагом в написании тестов для реализаций протоколов SMTP, POP3 и IMAP4 было изучение предметной области, отправление и получение писем напрямую из командных строк серверов. Затем были выделены требования из RFC и классифицированы по типам: команды и отклики, маршрутизация, уведомления, серверные настройки, заголовки письма, тело письма и прочие. На основании этого были построены упрощенные модели протоколов в которых автоматы тестов подавали в определенном порядке команды:

- для протокола SMTP: EHLO, HELO, MAIL, RCPT, DATA и другие;
- для протокола POP3: USER, PASS, LIST, STAT, RETR, DELE, TOP и другие;
- для протоколаIMAP4: тег плюс командное слово LOGIN, EXAMINE, CREATE, DELETE, RENAME, SELECT и другие,

а затем принимали отклики серверов: для протокола SMTP – трехзначные числа (коды откликов); для POP3 – "+OK" или "-ERR" отклики; для IMAP4 – необязательный тег плюс "OK", "NO", "BAD", "PREAUTH" или "BYE". На этом шаге реализация генерации тестовых наборов имела формальный интерфейс, спецификационные классы состояли только из сигнатур методов, все проверки корректности поведения тестируемых реализаций находились в сценарных классах. Сценарные классы состояли из методов, подающих воздействия целевым системам (посредством медиаторных классов), считывающих ответы серверов и возвращающих вердикты о корректности поведения серверов. Медиаторные классы переводили стимулы из формата тестируемой реализации в формат модели и обратно.

Затем были выделены основные состояния протокола. На данном шаге в сценарные классы были добавлены новые блоки, ответственные за перевод системы между состояниями. Спецификационные классы не менялись.

На следующем шаге блоки, ответственные за проверку корректности поведения тестируемой системы, и блоки, переводящие систему между состояниями, были перенесены из сценарных классов в спецификационные. В сценарных классах остались только стимулы, подающиеся тестируемой реализации. С этого момента определенные команды могли подаваться только из разрешенных состояний автомата. Возможность такой проверки

достигается записью разрешенных состояний в предусловия спецификации. Итератор каждый раз проверяет, в каком состоянии находится система, и разрешена ли подача данной команды в текущем состоянии. Также это позволяет проверить тот факт, что сервер не посылает команды из неразрешенных для данных команд состояний.

6.2. Тестирование взаимодействия реализаций протоколов

Для интеграционного тестирования реализаций протоколов создавался отдельный сценарий, который использует спецификации и медиаторы, разработанные для тестирования соответствия. Сценарий может содержать как сценарные методы, разрабатываемые для тестирования соответствия, так и модифицированные или специально разработанные непосредственно для тестирования интеграции. В частности, сценарии, не содержащие ветвления, могут быть просто скопированы в новый сценарный класс, в то время как сценарии, проверяющие реакцию протокола на различные параметры в командах, следует модифицировать, чтобы не проверять в интеграционном тесте несущественные для интеграции детали. Например, один из разработанных сценариев содержит следующие действия: для SMTP – последовательность команд для отправки письма, для POP3 и IMAP4 – последовательность команд для проверки количества писем в почтовом ящике, а также для чтения нового письма и сравнения его с письмом, отправленным в этом же тесте через протокол SMTP.

7. Анализ предложенного подхода

К недостаткам метода, основанного на формальных спецификациях, можно отнести отсутствие быстрой возможности обновления тестовых наборов. Для разработки нового теста необходимо тщательно изучить требования, формализовать и классифицировать их. Только после этих приготовлений можно начинать писать спецификационный, медиаторный и сценарный классы. Из-за стадии приготовления, включающей разработку спецификации, период разработки нового теста увеличивается. С другой стороны, после того, как написаны спецификационные, медиаторные и сценарные классы получается не один тест, а набор тестов, отвечающий за определенный класс требований. Также следует отметить, что длительная стадия приготовления присутствует только в случае неполной формальной спецификации. Если формальная спецификация доступна, тесты могут быть сгенерированы быстро, просто путем изменения сценарного класса. Более того, если задача состоит в написании сложных тестов, то использование формальных спецификаций даст результат быстрее и проще, чем в случае подхода, основанного на элементарных испытаниях.

Основной недостаток предложенного метода заключается в обязательности существования хотя бы одной реализации: на каждом шаге тесты и спецификация проверяются на реальной реализации. Как результат, этот

метод нельзя применять напрямую для протоколов, для которых еще нет какой-либо реализации.

Одним из важных достоинств этого метода является отделение блока, ответственного за вынесения вердикта, от блока генерации тестовых последовательностей. Оракул, который генерируется из постусловий спецификации, ответственен за вынесение вердикта. Благодаря этому разделению разработчик тестов не должен обрабатывать вердикты во время проверки корректности поведения системы. Наличие определенного блока (спецификации), в котором производятся и хранятся все проверки корректности поведения системы, упрощает переиспользование оракула в различных тестовых сценариях.

Также использование формальных спецификаций позволяет сформулировать точный недвусмысленный критерий полноты тестирования – тестирование может быть завершено, когда все элементы соответствующей формальной спецификации покрыты. UniTESK предоставляет поддержку динамического доступа к покрытым требованиям и позволяет задавать критерии отбора для сценариев – если применение конкретного сценария не увеличивает тестового покрытия, то система пропускает данный сценарий и переходит к следующему.

8. Результаты и дальнейшие исследования

В данной работе рассматривались только основные функции протоколов, другая функциональность, например, уведомления о доставке или сбоях, почтовые шлюзы в не-SMTP домены, выходят за рамки данного проекта. Для протокола SMTP было выделено 51 требование, 43 требования относятся к серверным командам и откликам (11 из них обязательные и 4 опциональные), 8 относятся к маршрутизации (все обязательные). Для протокола POP3 было выделено 58 требований на основные команды, 5 из них обязательные и 6 опциональные. Для протокола IMAP4 было выделено 43 требования на основные команды, 7 из них обязательные и 2 опциональные.

Все требования покрыты разработанными тестовыми наборами. Тесты применялись для почтовых серверов, разрабатываемых по модели открытого исходного кода – Apache James, hMail Server, Postfix и Dovecot. Сгенерированные тесты обнаружили несколько несоответствий между реализациями протоколов и стандартами [1-3]. При проведении интеграционного тестирования между реализациями протоколов SMTP и POP3 ошибок не обнаружено. Несоответствия, обнаруженные при тестировании соответствия, приведены ниже:

- отсутствие поддержки обязательных команд в отдельных реализациях;
- нарушение правил протокола (подача команд в недопустимых состояниях);

- неверные значения кодов откликов на команды протокола;
- отсутствие блокировки почтового ящика при аутентификации на сервере;
- зацикливание при пересылке сообщения.

8.1. Дальнейшие исследования

Общая характеристика почтовых протоколов заключается в том, что почтовые протоколы расширяемые. Некоторые расширения просто добавляют новую функциональность, то есть добавляют новые требования, которые не противоречат требованиям основного стандарта. Но также существуют расширения, которые радикально меняют структуру протокола, отвергая или изменяя отдельные требования базового стандарта. Тестирование расширяемых протоколов требует расширить инструментарий UniTESK средствами, которые предоставят возможность формально специфицировать расширения и генерировать тестовые последовательности с учетом набора расширений, поддерживаемых тестируемой реализацией. Направление нашего дальнейшего исследования – разработать соответствующие программные средства на базе инструмента JavaTESK.

9. Заключение

В статье представлен подход для тестирования почтовых протоколов, основанный на формальных спецификациях. Новизна метода заключается в последовательном переходе от тестов, написанных по методике элементарных испытаний, к формальной модели и тестам, сгенерированным с использованием этой модели как оракула. Более того, на каждом шаге получается работоспособный тестовый набор, проверяющий ту же функциональность, что проверялась на предыдущем шаге, или даже содержащую больше проверок.

Подход относится к области тестирования на основе моделей, он использует контрактные спецификации для формализации спецификаций протокола и генерации тестовой последовательности «на лету». Реализация подхода базируется на технологии UniTESK. Отличительными чертами данного метода являются автоматизированная генерация тестовой последовательности на основе формальных спецификаций, подсчет покрытия тестов, позволяющий оптимальным образом генерировать стимулы, а также наличие отдельного компонента – оракула – ответственного за вынесение вердикта о корректности поведения тестируемой системы.

Разработанный метод применялся для тестирования давно используемых реализаций почтовых протоколов. В одной реализации (Apache James Server) была найдена критическая ошибка – при определенной конфигурации DNS сервера во время пересылки сообщения сервер пересылал сообщение себе,

таким образом, сообщение никогда не достигало адресата. При этом уведомление отправителя о том, что сообщение не доставлено, отсутствует. Разработчики James признали выявленную проблему и планируют ее исправление в одном из ближайших релизов.

Литература

[1] IETF RFC 5321. J. Klensin. Simple *Mail Transfer Protocol*. 2008, 95 с.

[2] IETF RFC 1939. J. Myers, M. Rosem, *Post Office Protocol — Version 3*. 1996, 23 с.

[3] IETF RFC 3501. M. Crispin. *Internet Message Access Protocol – version 4rev1*. 2003, 108 с.

[4] ISO/IEC 9646. *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 1: General concepts.* Geneva: ISO (1994).

[5] *Apache James Mail Protocol Tester*. http://james.apache.org/mpt/antlib/

[6] Utting, M., Legeard, B.: Practical Model-Based Testing: A Tools Approach. Morgan Kaufmann, San Francisco (2007).

[7] Blackburn, M., Busser, R., Nauman, A.: *Why Model-Based Test Automation is Different and What You Should Know to Get Started*. Software Productivity Consortium, NFP (2004).

[8] Dalal, S.R., Jain, A., Karunanithi, N., Leaton, J.M., Lott, C.M., Patton, G.C., Horowitz, B.M.: *Model-Based Testing in Practice*. In: Proceedings of the ICSE 1999 (May 1999).

[9] ETSI ES 201 873-1 V3.1.1. *Methods for Testing and Specifcation (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language*. Sophia-Antipolis, France: ETSI (2009).

[10] Библиотека для тестирования программного обеспечения на языке Java. http://www.junit.org

[11] Jacky, J., Veanes, M., Campbell, C., Schulte, W.: *Model-based Software Testing and Analysis with C#.* Cambridge University Press, Cambridge (2008)

[12] Margus Veanes, Colin Campbell, Wolfgang Grieskamp, Wolfram Schulte, Nikolai Tillmann, Lev Nachmanson. *Model-Based Testing of Object-Oriented Reactive Systems with Spec Explorer*. Microsoft Research, Redmond, WA, USA. http://research.microsoft.com/pubs/77383/bookChapterOnSE.pdf

[13] Тестирование на основе моделей с помощью инструмента Spec Explorer. http://research.microsoft.com/en-us/projects/specexplorer/

[14] *End-to-End Testing Automation in TTCN-3 environment using Conformiq Qtronicand Elvior MessageMagic* (2009).

[15] Баранцев А.В., Бурдонов И.Б., Демаков А.В., Зеленов С.В., Косачев А.С., Кулямин В.В., Омельченко В.А., Пакулин Н.В., Петренко А.К., Хорошилов А.В. *Подход UniTesK к разработке тестов: достижения и перспективы*. // Труды ИСП РАН, №5, 2004. http://www.citforum.ru/SE/testing/unitesk

[16] UniTESK: индустриальная технология надежного тестирования. http://www.unitesk.com

[17] Н.В. Пакулин, А.В. Хорошилов. Разработка формальных моделей и тестирование соответствия для систем с асинхронными интерфейсами и телекоммуникационных протоколов. // Журнал "Программирование" № 5, 2007 г., ISSN 0132-3474, с. 1-29.

[18] V. V. Kuliamin, A. K. Petrenko, N. V. Pakoulin, A. S. Kossatchev, I. B. Bourdonov. Integration of Functional and Timed Testing of Real-time and Concurrent Systems. // Proceedings of the 5-th International Conference on Perspectives of System Informatics, July 9-12, 2003, Novosibirsk, Russia; LNCS 2890, Springer, 2003, pp. 450-461.

[19] V. V. Kuliamin, A. K. Petrenko, N. V. Pakoulin. Practical Approach to Specification and Conformance Testing of Distributed Network Applications. // Proceedings of the 2-nd International Service Availability Symposium, April 25-26, 2005, Berlin, Germany; LNCS 3694, Springer, 2005, pp. 68-83.

[20] V. V. Kuliamin, A. K. Petrenko, N. V. Pakoulin. Extended Design-by-Contract Approach to Specification and Conformance Testing of Distributed Software. // Proceedings of the 9-th World Multiconference on Systemics, Cybernetics, and Informatics, Model Based Development and Testing Workshop, July 10-13, 2005, Orlando, Florida, USA, pp. 65-70

[21] В. П. Иванников, А. С. Камкин, В. В. Кулямин, А. К. Петренко. Применение технологии UniTesK для функционального тестирования моделей аппаратного обеспечения. // Препринт 8 ИСП РАН, 2005.

[22] Иванников В. П., Камкин А. С., Косачев А. С., Кулямин В. В., Петренко А. К. Использование контрактных спецификаций для представления требований и функционального тестирования моделей аппаратуры. // Программирование, том 33, №5. МАИК «Наука/Интерпериодика», 2007, с. 47-61.

[23] I. B. Bourdonov, A. S. Kossatchev, V. V. Kuliamin, A. K. Petrenko. UniTesK Test Suite Architecture.// Proceedings of the International Symposium of Formal Methods Europe, July 22-24, 2002, Copenhagen, Denmark; LNCS 2391, 2002, pp. 121-152.

[24] I. B. Bourdonov, A. V. Demakov, A. A. Jarov, A. S. Kossatchev, V. V. Kuliamin, A. K. Petrenko, S. V. Zelenov. *Java Specification Extension for Automated Test Development.* // Proceedings of the 4-nd International Andrei Ershov Memorial Conference Perspectives of System Informatics, July 2-6, 2001, Novosibirsk, Russia; LNCS 2244, 2001, pp. 301-307.

[25] В. В. Кулямин, Н. В. Пакулин, О. Л. Петренко, А. А. Сортов, А. В. Хорошилов. Формализация требований на практике. // Препринт 13 ИСП РАН, 2006.