УДК 004.5; 004.8; 007.85

И.В. ГУРИНА, А.Я. КУЗЕМИН

МИНИМИЗАЦИЯ РИСКОВ ДЛЯ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ РАСПРЕДЕЛЕННОЙ ЭНЕРГЕТИЧЕСКОЙ СЕТИ ПОТРЕБИТЕЛЕЙ

Анализируются и описываются средства автоматизации тестирования сложных интеллектуальных систем. Предлагается методика определения наиболее эффективного варианта тестирования автоматизированной системы распределенной энергетической сети потребителей и производителей электроэнергии, позволяющая в лабораторных условиях эмулировать сложное окружение реального мира телекоммуникационных, клиент-серверных и Internet-взаимодействий и выполнять всестороннее тестирование распределенной системы. Предлагается метод покрытия ошибок системы с помощью математического моделирования границ транзакций. Ключевые слова – умные сети, принятие решений, selenium, chrome driver, надежность, автоматизированное тестирование, автоматизация запросов выполнения задач, Jenkins, транзакция.

1. Введение

Сфера энергетики обеспечивает жизнедеятельность и безопасность как всего общества в целом, так и каждого государства в отдельности, при этом технологической основой каждой энергосистемы являются электрические сети. Преобразование электроэнергетического комплекса в интеллектуальные сети является результатом многолетней научно-исследовательской деятельности, именно им сегодня уделяется особое внимание энергетиков.

В настоящее время появляются новые технологии производства электроэнергии – растет доля регенеративной электроэнергии, разрабатываются новые устройства, такие как электромобили, развиваются интеллектуальные электроэнергетические системы с активно-адаптивной сетью, построенной на основе использования новых принципов и технологий передачи и преобразовании электроэнергии. С появлением регенеративных источников энергии, таких как энергия ветра и воды, производство электроэнергии становится все более децентрализованным и более зависящим от погодных условий.

В связи с этим электрические сети становятся все более динамическими. Внедрение новых методов управления распределением энергии в сети требует широкого спектра использования соответствующих технологий. Одним из методов динамического управления потреблением энергии является управление спросом, задача которого – адаптация потребления энергии пользователями к ее фактическому производству.

В концепции «умной энергетики» основополагающим есть совпадение во времени процессов производства и потребления электроэнергии и равенство объема выработанной и потреблённой электроэнергии в каждый момент времени. Кроме внедрения новых устройств, одной из основных задач является разработка новой концепции управления такими “умными” электрическими сетями. Поэтому актуальны разработка и поддержка программного обеспечения, которое позволяет простое и гибкое конфигурирование, мониторинг и тестирование качества работы всех компонентов интеллектуальной среды в умных сетях Smart Grid.

2. Компьютерная сеть систем Smart Grid

Задача «Smart Grid» состоит в том, чтобы сделать «интеллектуальными» генерацию, передачу и распределение электроэнергии, обеспечить электрические сети современными диагностическими средствами, алгоритмами, электронными системами управления, техническими устройствами типа ограничителей токов короткого замыкания сверхпроводящих линий и множеством другим инноваций, появившихся на сегодняшний день в науке и технике. Актуальным является объединение возможностей информационных технологий, уже привычных для пользователей Интернет, с силовой электротехникой, что значительно

сократит потери при передаче электрической энергии от генератора к потребителю [4]. Многократное увеличение надежности энергоснабжения дает возможность оптимально перераспределять энергетические ресурсы и тем самым уменьшать пиковые нагрузки (все электротехнические системы проектируются именно в расчете на пиковые нагрузки). Это позволяет потребителю быть активным игроком на рынке электроэнергии. Ведь если раньше потребитель брал электрическую энергию от одного продавца, то теперь он находится в условиях рынка: может выбирать среди конкурирующих производителей.

Разработка средств эффективного взаимодействия человека с компьютером считается одним из приоритетных направлений развития искусственного интеллекта и информатики в целом.

Разработка человеко-машинных интерфейсов для веб-приложений и динамическое взаимодействие пользователя с интеллектуальной системой имеет огромное значение в системах управления, так как они должны отслеживать и контролировать реагирование физических систем на различные условия окружающей среды в режиме реального времени. Таким образом, крайне важно, чтобы оператор системы управления мог принимать огромное количество структурированной информации о физическом состоянии системы, при этом принимать решения в кратчайший период времени.

В интеллектуальной электроэнергетической системе задача визуализации работы всех компонентов является более важной, чем в традиционной электроэнергетике, так как конечные потребители активно участвуют в работе системы в режиме реального времени и поэтому крайне важно обеспечить стабильность и корректность работы всех компонентов энергетического комплекса.

3. Анализ систем безопасности Smart Grid

Одним из важных составных аспектов безопасности информации в “умной сети” Smart Grid является надежность системы, в частности - отказоустойчивость в критических и чрезвычайных ситуациях. Угроза безопасности работы компонентов “умной сети” Smart Grid может возникать не только в результате преднамеренных атак на информационную систему или неправильных действий пользователя, но также и из-за превышения критической нагрузки на систему. Для надежного и качественного функционирования интеллектуальной системы необходимо иметь данные о предельной нагрузке (выраженной, например, в количестве синхронно поступающих в систему запросов пользователя), о характере поведения системы в условиях перегрузки и возможных последствиях в случае ее отказа при превышении допустимой нагрузки [3].

Интеллектуальные системы, использующие математические модели и программные алгоритмы тестирования, могут выполняться в минимальной конфигурации технических средств. Применение в интеллектуальных системах знаний об объекте управления в виде формализованных описаний его свойств позволяет создавать информационную избыточность для значимых переменных решаемых задач, которая собственно и используется для автоматизированного тестирования [5], [6]. В этом случае повышение отказоустойчивости происходит без увеличения количества датчиков первичной информации, т.е. исключаются многие проблемы, присущие аппаратному тестированию.

4. Постановка задачи

Интеллектуальная сеть Smart Grid обеспечивает активное управление, контроль и участие как производителей, так и потребителей электрической энергии.

Для обеспечения качественной работы «умной сети» необходимо наложение коммуникационной инфраструктуры и соответствующих сервисов на энергосистему, по сути, необходимо соединить все компоненты в единую коммуникационную систему в Интернете, таким образом, чтобы коммуникация происходила в режиме реального времени.

Для Smart Grid используется множество различных технологий, протоколов, интерфейсов и иногда запатентованные технологии. Центральным фактором работы “умной сети” в реальных условиях является качественное взаимодействие всех компонентов.

Актуальным является разработка инструментальных средств, позволяющих в лабораторных условиях эмулировать сложное окружение реального мира клиент-серверных, телекоммуникационных и Internet-взаимодействий и выполнить всестороннее тестирование

распределенной системы. Задача таких инструментальных средств — организация лабораторного испытательного стенда, который будет эмулировать от десятков до тысяч пользователей, посылающих и получающих информацию, воспроизводя тем самым сложное взаимодействие между клиентским и серверным приложением, базами данных, Internet-серверами и другими системами.

5. Автоматизированное тестирование сложных систем

Облачные технологии – это модель предоставления повсеместного и удобного сетевого доступа к общему пулу конфигурируемых вычислительных ресурсов. На сегодняшний момент многие организации внедряют виртуализацию и облачные технологии в сложные интеллектуальные системы для снижения затрат и повышения эффективности. Виртуализация и облачные инфраструктуры также используются для расширения автоматизированного тестирования за счет сокращения инвестиций в физическое оборудование, необходимое для создания тестовой среды. Внедрение этих технологий является кросс-браузерным и может быть использовано для различных операционных систем. Selenium WebDriver имеет полную техническую поддержку для тестирования приложений в виртуальной среде, а кроме того представляет возможность выполнять тесты параллельно, что снижает затраты и повышает скорость и мобильность тестирования сложной распределенной системы.

6. Предлагаемое решение

К основным задачам автоматизированного тестирования относятся повышение критической надежности, включая самые слабые и уязвимые звенья, упреждение системных рисков и угроз, быстрое реагирование на инциденты и аварии. Вот смысл, который заложен в автоматизированное тестирование «умных сетей» или «Smart Grid». Selenium - надежный фреймворк автоматизации, способный работать с любым браузером, позволяющий разрабатывать тесты с очень сложной логикой поведения и проверок. Selenium Server может принимать команды с удалённой машины, где работает сценарий автоматизации, и исполнять их в браузере. Несколько серверов Selenium могут образовывать распределённую сеть, которая называется Selenium Grid, что позволяет легко масштабировать стенд автоматизации. Selenium-Grid используется для ускорения выполнения тестового прохода с помощью нескольких компьютеров для выполнения тестов параллельно. Например, если имеется тестовый набор, состоящий из 100 тестов, но при этом Selenium-Grid распределена на четырех различных серверах (виртуальные машины или отдельные физические машины), прохождение тестов будет производиться в четыре раза быстрее по сравнению с тем, если бы тесты запускались последовательно на одной машине.

Для больших наборов тестов и комплексных тестов, имеющих большие объемы данных для проверки, это может стать значительной экономией времени. Ведь для выполнения некоторых тестов может понадобиться несколько часов.

Кроме того, быстрое выполнение тестов позволяет сократить время для устранения той или иной ошибки, что очень важно в таких сложных комплексных системах как Smart Grid.

Вместе с тем, автоматизированные тесты позволяют получать немедленную обратную связь.

Общая структура интеллектуальной электроэнергетической системы крайне разнородна по широкой совокупности технико-технологических и иных параметров и характеристик, что замедляет процесс ликвидации существующих ошибок в работе и требует упорядочения в рамках единого подхода, который должен быть целесообразно сформирован на новых принципах построения автоматических тестов. На рис. 1 представлена предлагаемая модель системы.

Для объективного оценивания свойства отказоустойчивости интеллектуальной системы мы предлагаем использовать следующий показатель:

$$\varphi = \sum_{i=1}^{N} f_i(c_i) * (f_{max} * N)^{-1}, \quad (1)$$

где f_i – число функций, корректно выполняемых интеллектуальной системой в состоянии c_i (например, в случае отказа работы какого-либо компонента интеллектуальной систе-

мы); f_{max} – число функций, которые выполняет полностью исправная система; N – количество возможных состояний интеллектуальной системы.

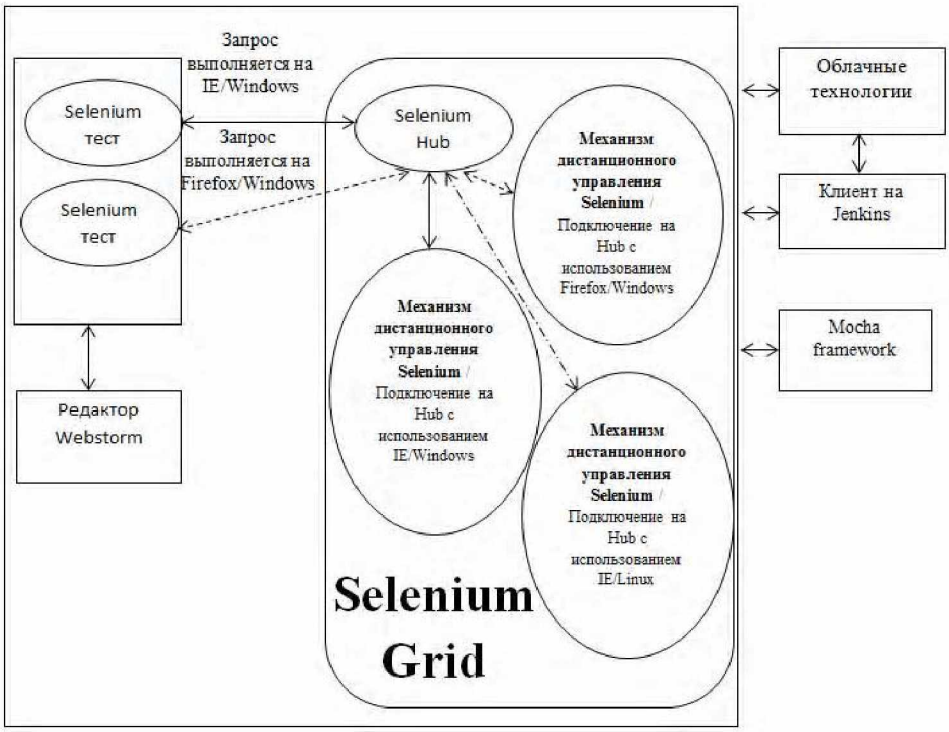

Рис. 1. Модель тестирования качества работы интеллектуальной энергетической системы

Одной из новейших разработок в области тестирования, которая соответсвует требованиям интеллектуальной системы Smart Grid, является сервис „Sauce On Demand“ (рис. 2).

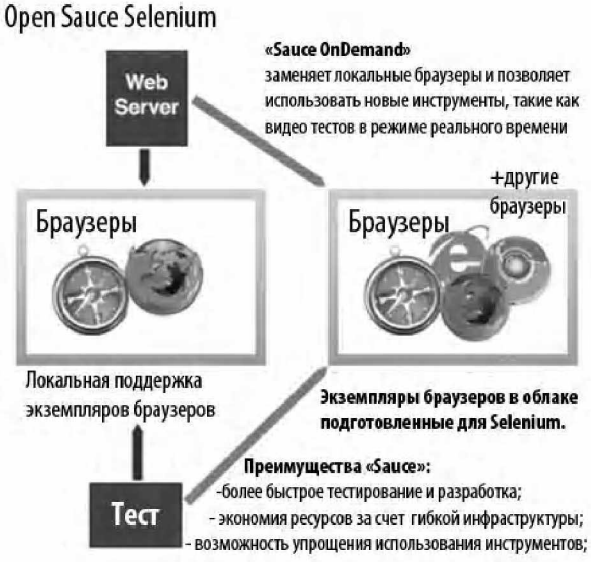

Рис. 2. Сервис „Sauce On Demand“

Данный продукт построен на основе продукта Selenium, но в отличие от него требует меньших временных ресурсов. Кроме того, данный сервис выполняет непрерывное сканирование экрана в каждой тестовой машине и хранит все видео в Amazon S3. Это позволяет выполнять Selenium кросс-браузерные тесты в облаке без необходимости настройки среды тестирования.

Amazon Simple Storage Service (Amazon S3) — онлайновая веб-служба, предлагаемая Amazon Web Services, предоставляющая возможность для хранения и получения любого объёма данных, в любое время из любой точки сети, так называемый файловый хостинг. На сегодняшний день было запущенно в общем числе 3 миллиона тестов, все из них были конвертированы в видео.

В качестве фреймворка тестирования мы предлагаем использовать JavaScript Mocha, который является многофункциональным фреймворком JavaScript, позволяет работать отдельно с узлами и браузерами и выполнять асинхронное тестирование просто и качественно. Mocha-тесты выполняются последовательно, что обеспечивает более гибкую возможность выполнения точной отчетности.

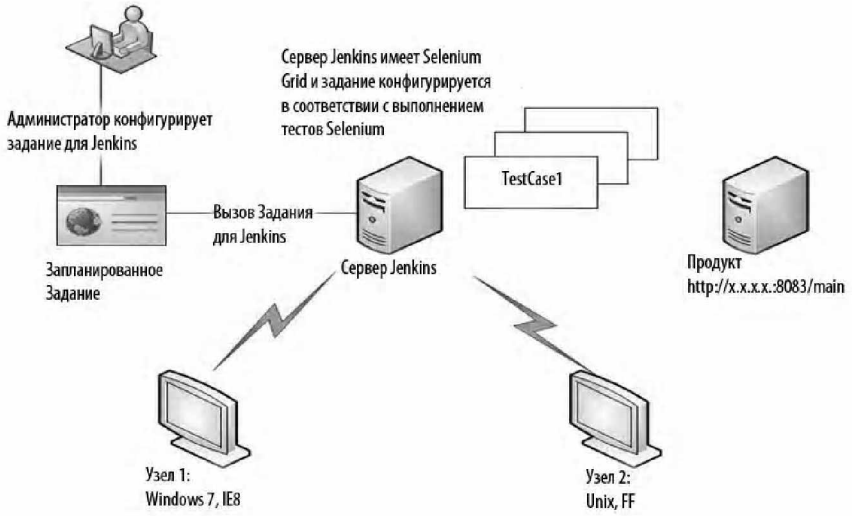

Рис. 3. Принцип работы с сервером Jenkins

Кроме того, мы используем сервер непрерывной интеграции Jenkins (рис. 3), который позволяет проверить, что каждая сборка проекта прошла успешно. Результаты анализа системы отображены на диаграме (рис. 4).

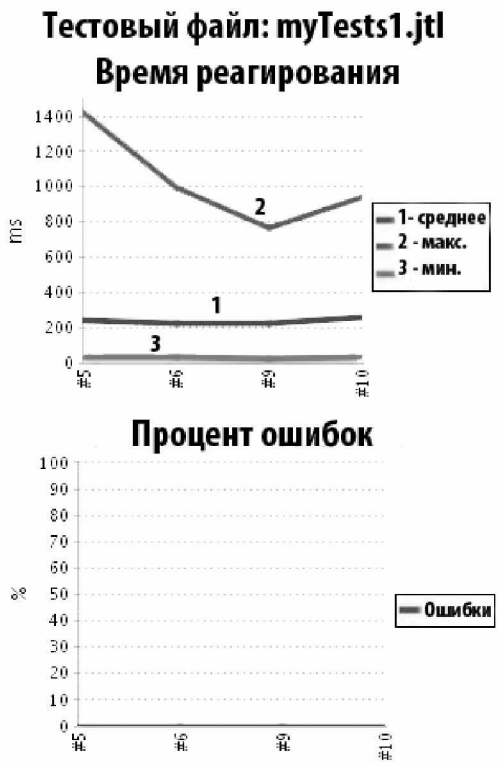

Рис. 4. Диаграмма ошибок в Jenkins

А если при сборке возникли какие-то проблемы, то оператор немедленно об этом узнает, - ему будет отправлено уведомление (рис. 5). И кроме того, при использовании сервера непрерывной интеграции не нужны специальные ручные действия. Сервер CI сразу после изменений сам выполнит все тесты проекта, запустит анализаторы кода и соберет различные метрики. Кроме того, Jenkins поволяет осуществлять нагрузочное тестирование с помощью интеграции технологии JMeter.

JMeter является эмулятором работы большого количества пользователей (рис. 6).

JMeter дает возможность генерировать количество пользователей (number of threads), время, за которое начнут работу все пользователи (ramp-up-period), количество циклов для каждого пользователя (loop count).

Для каждого виртуального пользователя JMeter порождает нить. С помошью последовательности HTTP-запросов задается сценарий поведения пользователя («план тестирования»). На данном этапе развития информационных технологий отсутствует возможность со-

здавать сценарии поведения пользователей при помощи браузера. Управление запросами осуществляется с помощью модулей, называемых контроллерами, а доступ к данным тестирования – при помощи слушателей. Каждый слушатель собирает информацию, относящуюся только к одной определенной группе виртуальных пользователей. Собранную информацию можно представить в текстовой форме с помощью слушателя File Reporter или в виде графика с помощью слушателя Graph Results. При этом производится и первичная статистическая обработка результатов. Интервалы времени между запросами задаются с помощью таймеров.

All +

S	W	Name ↓	Крайний успех	Крайняя неудача	Крайняя продолжительность	
		check that customer is online	Неизвестно	Неизвестно	Неизвестно	
		recent customer activity check	Неизвестно	13 дней - #2	13 секунд	
		gas meter check	Неизвестно	Неизвестно	Неизвестно	
		electricity meter check	Неизвестно	Неизвестно	Неизвестно	
		check the message for the customer	13 дней - #2	Неизвестно	5 минуты 27 секунд	
		peak load testing	Неизвестно	13 дней - #1	53 секунд	
		network analysis	7 дней 19 часов - #106	4 дня 21 часов - #113	4 минуты 20 секунд	
		demand response optimization testing	15 дней - #4	Неизвестно	17 секунд	
		customer segmentation testing	1 месяц 3 дня - #4	Неизвестно	30 минут	

Иконка: S M L

Легенда RSS для всех RSS для неудачных RSS для последних сборок

Рис. 5. Пример реализованного теста проверки отправки сообщений пользователям

Одним из достоинств JMeter является возможность удаленного тестирования с использованием клиент-серверной архитектуры на основе механизма Java RMI. Генерация нагрузки и сбор данных осуществляются на одном узле сети, а обработка и визуализация данных – на другом, что минимизирует влияние параметров среды передачи данных на результаты измерений. Клиент-серверная архитектура дает возможность построения распределенной системы тестирования путем размещения независимых тестирующих компонентов JMeterEngines на узлах вычислительной сети и удаленного управления ими.

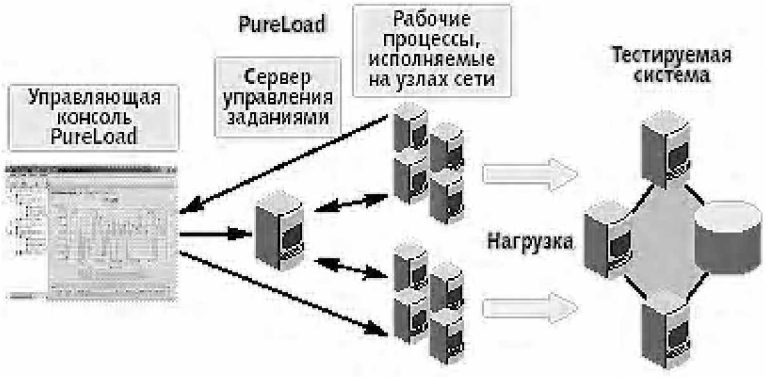

Рис. 6. Конфигурация работы системы Jmeter

Еще одно достоинство JMeter, упрощающее обработку результатов тестирования, – это возможность использования тестовых утверждений (assertion). Ее наличие позволяет автоматически проверять соответствие ответов сервера, полученных в ходе тестирования, ожидаемым величинам. Например, с помощью тестовых утверждений, записанных в форме регулярных выражений, можно проверить наличие определенного текста в HTTP-

ответе. Использование Java не только обеспечивает поддержку большого числа платформ и облегчает построение распределенных систем, но также способствует расширяемости данной системы тестирования.

Развитию JMeter способствует и доступность исходных текстов; это позволяет надеяться на повышение качества данного инструментария в будущем.

Результат теста – максимальное число пользователей, которые могут одновременно получить доступ к веб-узлу, число запросов, обрабатываемых приложением, или время ответа сервера.

Основываясь на полученном результате, веб-мастер и сетевой администратор (в работе сервера участвуют и другие компоненты сети, маршрутизаторы, брандмауэр, кэширующий и прокси-сервер, база данных) смогут заранее выявить узкие места, возникающие из-за несбалансированной работы компонентов, и исправить ситуацию, перед тем как включать систему в реальную работу.

Во время тестирования имитируется одновременная работа нескольких сотен или тысяч посетителей. Для большей правдивости каждый из виртуальных пользователей может «ходить» по сайту по индивидуальному сценарию и иметь личные параметры. Также в процессе тестирования можно имитировать кратковременные пики нагрузки, когда количество посетителей скачкообразно увеличивается, что очень актуально для сайтов с неравномерной аудиторией.

На рис. 7 представлен пример реализованного теста проверки отправки сообщений пользователям. В данном тесте мы объявляем переменную пользователя, затем в отдельной переменной объявляем текст, который хотим отправить, после этого заходим в Web-приложение как определенный пользователь, затем по ссылке переходим в диалоговое окно, отправляем сообщение и после этого проверяем, что сообщение было доставлено.

```
it ('should be able to send dialog message', function () {
var participant= users.StephenPowers,
textToSend = 'Hello', +participant.name + '! Current UNIX
time is' + (new Date()).getTime();
helper.loggedInAs(users.Andrew.Boyd);
browser.get('/messages/dialogs/id:' +participant._id.toString());
sendDialogMessage(textToSend);
lastDialogMessage('Hello').should.be.equal(textToSend);
});
```

Рис. 7. Пример проверки отправки сообщений пользователям

```
helper.loggedInAs(users.AndrewBoyd);
browser.get('/messages/SendSms');
selectReceiver(groupName);
typeMessage(textToSend);
sendMessage();
selectDialog(users.Trishamiller.name);
lastDialogMessage().should.be.equal(textToSend);
selectDialog(users.WendieLarkin.name);
lastDialogMessage().should.be.equal(textToSend);
selectDialog(users.WendieLarkin.name);
lastDialogMessage().should.be.equal(textToSend);
```

Рис. 8. Читаемый тест

Для поиска необходимого элемента в браузере мы используем XPath. Xpath — это язык запросов к элементам xml или xhtml документа. Также как SQL, xpath является декларативным языком запросов. Чтобы получить интересующие данные, необходимо всего лишь создать запрос, описывающий эти данные. Например, нам необходимо получить элемент ссылки:

```
<div>
 <div id="div-id">некоторый div</div>
 <div>еще один div</div>
 <a href="http://internetka.in.ua/">
 <img src="url" alt="картинка-ссылка" />
 </a>
</div>
```

Для этого достаточно написать один из представленных ниже запросов:

```
driver.findElement(By.xpath("//div[@id='div-id']/..//a"));
driver.findElement(By.xpath("//div[@id='div-id']/following-sibling::a"));
```

Кроме того, с помощью XPath (и DOM) можно найти элемент относительно расположения других элементов на этой странице. Например, если необходимо проверить, что данная ссылка находится во втором параграфе текста, внутри секции . Это можно задать при помощи XPath. При использовании локаторов ID и name можно проверить лишь факт наличия элемента на странице вообще, без указания определенного местоположения. Например, если есть необходимость проверить, что элемент отображается наверху сайта, в секции <header>, то следует использовать локатор XPath. Была разработана методика, согласно которой внутренняя структура всех функций инкапсулируется в отдельный файл, тем самым мы упрощаем структуру тестов и делаем их более читабельными. Здесь подразумевается то, что сторонний читатель, обладающий исключительно контекстом предметной области, в которой написан тест, читая только тест, не вынужден отвлекаться на выяснение деталей в другой контекст, может интуитивно догадаться о сути теста и выполняемых им утверждениях и проверках (рис. 8).

7. Покрытие ошибок системы путем тестирования переходов

Мы предлагаем метод покрытия ошибок системы с помощью математического моделирования границ транзакций [7, 8].

Предлагается использовать шестерку:

$$A = (I, O, L, l_0, C, T) \ ,$$

где I – конечное множество входов, которые представляют собой сообщения, полученные системой от окружающей среды; O – конечное множество выходов, которые представляют собой сообщения, передаваемые системой в окружающею среду; L – конечное множество положений. Положение представляет собой "статус" системы после выполнения перехода; $l_0 \in L$ – начальное положение системы; $T \subseteq L \times (I \cup O) \times \phi(C) \times P(C) \times L$ – множество транзакций.

Транзакция обозначенная, как $t : l \xrightarrow{m, G, R} l'$, состоит из начального положения l, входящего или выходящего сообщения m, временной границы G, подмножества счетчиков R, которые сбрасываются, когда транзакция выполнена, и целевого положения l' (т.е. $цель(t) = l'$). Каждый счетчик в R используется для записи, чтобы определить, сколько времени прошло после выполнения транзакции. Такие счетчики в основном применяются для определения временных границ между транзакциями, в момент их обнуления, и будущими транзакциями. Будем считать, что транзакции выполняются мгновенно (т.е. они не требуют времени для выполнения). Кроме того, транзакции на входах ограничены элементарными конъюнкциями в виде $x \, op \, b$, где $x \in C, op \in \{<, \leq, =, >, \geq\}$, а b – натуральное число, в то время, как транзакции на выходах ограничены элементарными конъюнкциями в виде $x \, op \, b$, где $x \in C, op \in \{\leq, =, \geq\}$ и b – натуральные числа. Администратору, тестирующему систему, необходимо знать точное время выходов, так как они являются

неконтролируемыми величинами. В определении множества транзакций Φ(C) и P(C) обозначают временные границы и мощность множества C, соответственно. Множество счетчиков используется для определения временных ограничений между более чем двумя транзакциями. Кроме того, мы предполагаем, что каждый счетчик $x \in C$ имеет ограниченную область $[0, B_x] \cup \{\infty\}$, где B_x – наибольшее целое число, являющееся ограничением по времени для x . Это означает, что каждый счетчик x является релевантным только в рамках целой константы B_x , и все значения x, больше чем B_x , представляются как ∞ . Таким образом, отсюда следует выражение:

$$\forall \varepsilon > 0, B_x + \varepsilon = \infty \text{ and } \forall \varepsilon > 0, \infty + \varepsilon = \infty \ .$$

Для счетчика x и для временной границы G определяется проекция G на x, обозначенная как $\mathrm{Proj}(G, x)$, по условию $(x \ \mathrm{op}\ b)$ удаляются все счетчики, кроме x; если счетчик не включен в G, то $\mathrm{Proj}(G, x) = \mathrm{true}$.

На рис. 9 показан пример, описывающий поведение простой мультимедийной системы. Система получает изображение, за которым следует звуковое сообщение в течение двух единиц времени, затем она посылает подтверждение, ackAll, не более чем через три единицы времени после получения изображения и не более чем через две единицы времени после того, как получено звуковое сообщение, затем сообщение сбрасывается и система возвращается в начальное положение в режим ожидания следующего изображения.

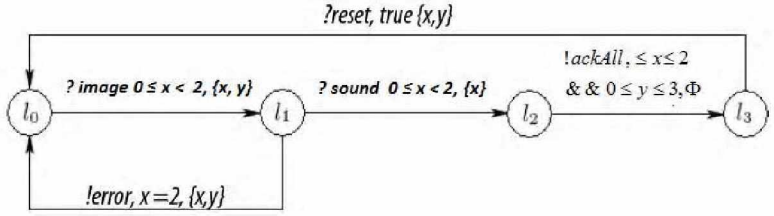

Рис. 9. Транзакции простой мультимедийной системы

Начальное положение системы l_0 . Транзакция $l_0 \xrightarrow{m, G, R} l_1$ выполняется, когда система получает сообщение с изображением и значение счетчика x меньше, чем 2. Когда транзакция выполнена, значения счетчиков x и y приравниваются 0. В данном примере временные области счетчиков, соответственно, $([0,2] \cup \{\infty\})$ и $([0,3] \cup \{\infty\})$. Проекция временной границы $(0 \le x < 2)$ на счетчик x равна $(0 \le x \le 2)$, пока выполняется проекция на счетчик y (т.е. счетчик y не включен во временную границу). Однако проекция временной границы $((0 \le x < 2) \,\&\&\, (1 \le y \le 2))$ на счетчик x равна $(0 \le x \le 2)$, в то время как проекция на счетчик y равна $(1 \le y \le 2))$ (здесь, оба счетчика x и y включены во временную границу).

8. Выводы

Высокое покрытие тестами является важным этапом обеспечения работы интеллектуальной системы Smart Grid.

Был предложен метод минимизации рисков для автоматизированной системы распределенной энергетической сети потребителей с помощью автоматизированных тестов.

Предложен метод покрытия ошибок системы с помощью математического моделирования границ, предотвращающий ошибки при выполнении транзакций.

Таким образом, предложенная методика определения наиболее эффективного варианта тестирования автоматизированной системы распределенной энергетической сети потребителей и производителей электроэнергии позволяет эмулировать участие в работе сети потребителей; может проверять устойчивость сети к физическому и кибернетическому

вмешательству злоумышленников; тестирует обеспечение синхронной работы источников генерации и узлов хранения электроэнергии; повышает эффективность работы энергосистемы в целом.

Список литературы: 1. *A. K. David and F. Wen*, Market Power in Electricity Supply, Proceedings of 2002 Power Engineering Society Winter Meeting. 2002. Vol. 1. P. 452-457. **2.** *J. B. Park, et al.*, A Continuous Strategy Game for Power Transactions Analysis in Competitive Electricity Markets," IEEE Trans. on Power Systems, 2000. Vol. 16. P. 847-855. **3.** *J. W. Bialek*, Gaming the Uniform-Price Spot Market - Quantitative Analysis," IEEE Trans. on Power Systems, 2002. Vol. 17. P. 768-773. **4.** *Kyzemin Oleksandr, Gurina Iryna*, Methods and models of the effective distribution of energy between consumers and producers, ITHEA, 2013. **5.** *Ларичев О.И.* Теория и методы принятия решений. М.: Логос, 2002. 392 с. **6.** *Питерсон Дж.* Теория сетей Петри и моделирование систем /Пер. с англ. М.: , Мир, 1984. 264 с. **7.** *A. En-Nouaary, R. Dssouli, and F. Khendek*, "Timed Wp-Method: Testing Real-Time Systems", IEEE *Transactions on Software Engineering*, 28(11). 2002. P. 1023–1038. **8.** *A. Khoumsi, A. En-Nouaary, R. Dssouli, and M. Akalay*, "A New Method for Testing Real-Time Systems", In Proc. of the Seventh International Conference on Real-Time Systems and Applications,Cheju Island, South Korea. 2000. P. 441–451.