УДК 004.056.23 Вестник УрФО № 4(38) / 2020, с. 55–63

Ручай А.Н.

DOI: 10.14529/secur200405

РАЗРАБОТКА НОВОГО ПОДХОДА РЕГРЕССИОННОГО ТЕСТИРОВАНИЯ С ПОЛУАВТОМАТИЧЕСКИМ ВЫБОРОМ ТЕСТОВ ДЛЯ АУДИТА IMS БАЗЫ ДАННЫХ

Данная работа посвящена разработке нового подхода регрессионного тестирования с полуавтоматическим выбором тестов. В отличие от предыдущих работ, предложенный подход основывается на полуавтоматическом выборе тестов с учетом различных важных факторов, таких как время выполнения, необходимое количество выполнений тестов, ручной приоритет тестов, результат предыдущих тестов, функциональные возможности тестов. Это позволяет более детально настроить последовательность регрессионных тестов в полуавтоматическом режиме.

Ключевые слова: *тестирование программного обеспечения, регрессионное тестирование, отладка, тестирование на основе ключевых слов, тестирование на основе данных, база данных IMS, выбор тестов.*

Ruchay A.N.

DEVELOPMENT OF A NEW REGRESSION TESTING APPROACH WITH SEMI-AUTOMATIC TEST SELECTION FOR AUDITING AN IMS DATABASE

This work is devoted to the development of a new regression testing approach with semi-automatic test selection. Unlike previous works, the proposed approach is based on semi-automatic selection of tests taking into account various important factors, such as execution time, the required number of test executions, manual priority of tests, the result of previous tests, and test functionality. This allows you to fine-tune the sequence of regression tests in semi-automatic mode.

Keywords: *software testing, regression testing, debugging, keyword driven testing, data driven testing, IMS database, test selection.*

1. Введение

Современные программные проекты развиваются быстро, поскольку разработчики добавляют новые функции, исправляют ошибки или проводят рефакторинг [1, 2, 3]. Чтобы гарантировать, что развитие проекта не нарушит существующую функциональность, разработчики обычно проводят регрессионное тестирование. Однако частый повторный запуск наборов полных регрессионных тестов может занять очень много времени. Для запуска некоторых наборов тестов требуются недели, но ожидание результатов тестирования даже в течение нескольких минут может нанести ущерб рабочему процессу разработки. Помимо снижения производительности, медленное регрессионное тестирование может потреблять много вычислительных ресурсов. В результате большое количество исследований было посвящено снижению затрат на регрессионное тестирование с использованием таких подходов, как выбор регрессионных тестов [4, 5], сокращение набора регрессионных тестов [6, 7], приоритезация регрессионных тестов [8, 9] и распараллеливание тестов.

Подход к регрессионному тестированию, который представлен в данной работе, отличается от других подходов тем, что он использует полуавтоматический процесс выбора тестов для определения приоритетности их выполнения. Приоритизация основана на различных критериях, таких как время выполнения, необходимое количество запусков тестов, ручной приоритет тестов, результат предыдущих тестов, функциональность тестов. Этот подход позволяет более детально настроить последовательность регрессионных тестов.

Работа организована следующим образом. В разделе 2 представлена информация о регрессионном тестировании системы аудита баз данных IMS. В разделе 3 описываются различные подходы к автоматизированному тестированию и автоматизации тестирования. Также в разделе 3 описана структура фреймворка с веб-интерфейсом. Раздел 4 посвящен изучению подхода к выбору регрессионного теста, который ускоряет регрессионное тестирование, а также представлен подход регрессионного тестирования с полуавтоматическим выбором тестов. Наконец, в разделе 5 представлены выводы.

2. Регрессионное тестирование системы аудита IMS базы данных

IBM Information Management System (IMS) – это объединенная иерархическая база данных и система управления информацией с обширными возможностями обработки транзакций [11, 12, 13].

Иерархические базы данных IMS имеют множество преимуществ: это проверенная технология, применяемая в течение многих лет; иерархия (древовидная структура) IMS баз данных проста в использовании. Базы данных IMS способны управлять более 15 миллионами гигабайт данными, обрабатывать транзакции от более 200 миллионов пользователей, и поддерживать более 22000 транзакций в секунду. Основным недостатком иерархических баз данных IMS является их сложность, как с точки зрения правил, так и с точки зрения отношений данных «многие ко многим». Эта сложность привела к тому, что работа с IMS оставалась очень медленной и громоздкой [11, 12, 13].

Система аудита для базы данных IMS на z/OS – это инструмент аудита, который собирает и сопоставляет информацию о доступе к данным из онлайн-регионов IMS, пакетных заданий IMS, наборов данных архивных журналов IMS и записей SMF для получения всестороннего обзора происходящей бизнес-активности в одной или нескольких средах IMS. z/OS – это операционная система для мэйнфреймов IBM, производимая IBM. Система аудита для базы данных IMS помогает аудиторам определить, кто читал или обновлял конкретную базу данных IMS и связанные с ней наборы данных, какой механизм использовался для выполнения этого действия и когда был произведен доступ.

Система аудита для базы данных IMS обеспечивает комплексное решение для аудита, которое снижает затраты на соблюдение нормативных требований за счет автоматизации, централизации и разделения обязанностей. Автоматизация упрощает сбор данных аудита, сокращает ручные усилия, увеличивает производительность, обеспечивает более тщательный аудит и снижает его стоимость. Система аудита для базы данных IMS централизует и сопоставляет информацию из многих систем и источников данных, чтобы обеспечить согласованное представление дан-

ных аудита IMS, создавать отчеты с графическим интерфейсом и пакетные отчеты для проверки данных, а также поддерживать внутренних и внешних аудиторов. Разделение обязанностей обеспечивает целостность данных аудита, устраняет возможность подделки данных, обеспечивает возможность аудиторов не зависеть от разработчиков или администраторов баз данных в настройке или сборе требуемой информации аудита.

Для тестирования системы аудита для баз данных IMS было использовано 50 онлайн-заданий z/OS и 50 пакетных заданий z/OS с 1000 транзакциями в секунду, 10 скриптов стресс-тестирования со 100000 транзакциями в секунду, 1000 транзакций IMS LOG и 3000 SMF транзакций. Наши наборы тестов включали 120 автоматических тестов, 350 тестовых случаев, 4 версии системы аудита для базы данных IMS, 4 версии баз данных IMS и 3 подсистемы z/OS.

Постоянное регрессионное тестирование (общее тестирование, тестирование безопасности, тестирование производительности, стресс-тестирование) необходимо, чтобы гарантировать, что добавление новых функций, исправлений или рефакторингов не нарушит существующую функциональность. Однако частый повторный запуск наборов полных регрессионных тестов требует очень много времени. Для выполнения этого тестирования могут потребоваться недели и это может нанести ущерб рабочему процессу разработки. Помимо снижения производительности труда разработчиков, медленное регрессионное тестирование может потреблять значительные вычислительные ресурсы. Нашему регрессионному тестированию требуется около 3 месяцев для запуска всех тестов вручную. Основная цель заключалась в создании среды автоматизации тестирования, которая сокращает время регрессионного тестирования.

3. Избирательная мультибиометрическая аутентификация

Автоматическое тестирование – это выполнение тестовых примеров в автоматическом режиме без ручного вмешательства [14]. Автоматизация тестирования включает в себя различные действия, такие как создание тестов, создание отчетов о результатах выполнения тестов и управление тестированием. Этот метод тестирования эволюционировал в течение пяти поколений [15, 16]: запись и воспроизведение, пользовательские скрипты, тестирование на основе данных, тестирование на основе ключевых слов, и тестирование, управляемое процессами.

Фреймворк автоматизации тестирования – это интегрированная система, которая устанавливает правила, по которым продукт проходит автоматическое тестирование. Среда автоматизации тестирования объединяет библиотеки функций, источники тестовых данных, сведения об объектах и различные повторно используемые модули. Фреймворк обеспечивает основу для тестирования и упрощает автоматизацию. Наиболее распространенные подходы автоматизации тестирования являются [15, 16]: тестирование на основе модулей; тестирование под управлением данными; тестирования под управлением ключевыми словами; и гибридное тестирование.

Тестирование на основе модулей – это базовый подход автоматизации тестирования. Платформа тестирования на основе модулей использует сценарии тестирования, которые соответствуют функциональности продукта и соответствуют конкретным модулям приложения. Эти тестовые сценарии используются иерархически для создания больших тестовых примеров. Структура тестирования на основе модулей назначает независимые тестовые сценарии конкретным программным компонентам, модулям или функциям.

В подходе автоматизации тестирования под управлением данными тестовые данные отделяются от тестовых сценариев, и результаты тестирования сравниваются с тестовыми данными. Когда тесты используют различные наборы входных и выходных тестовых данных, среды тестирования на основе данных более эффективны и проще в управлении.

В подходе автоматизации тестирования под управлением ключевыми словами ключевые слова написаны таким образом, чтобы они соответствовали функциональности на уровне модулей. Фреймворк тестирования на основе ключевых слов не зависит от приложения и использует методы таблицы данных и ключевые слова для выполнения действий.

Гибридное тестирование включают в себя модульные сценарии, фреймворк на основе данных и функциональная декомпозиция. Платформа гибридного тестирования может быть более сложной для первоначальной настройки, чем другие платформы. Одна-

ко инфраструктуры гибридного тестирования могут обеспечить максимальную гибкость, если они будут тщательно оценены и внедрены.

Используемая нами структура автоматизации тестирования состоит из системы проектирования тестов, системы мониторинга тестирования и системы выполнения тестов [17].

Веб-интерфейс, который мы использовали с нашей платформой автоматизации тестирования, был разработан с помощью Qooxdoo, среды веб-приложений Ajax с открытым исходным кодом. Это клиентское, независимое от сервера решение, которое включает поддержку профессиональной разработки на JavaScript [18], набор инструментов графического пользовательского интерфейса (GUI) и высокоуровневую связь между клиентом и сервером.

Библиотеки Java, которые мы использовали для разработки нашей среды автоматизации тестирования, включают: S3270 для интеграции системы TN3270 на z/OS, Selenium WebDriver для экземпляра браузера, Ganymed SSH-2 для реализации протокола SSH-2, Apache Commons Net для реализации на стороне клиента основных интернет-протоколов, Apache FreeMarker для генерации текстового вывода на основе шаблонов и изменения данных, Jsoup для работы с реальным HTML, Gson для сериализации и десериализации Java-объектов в JSON, Apache log4j для ведения журнала, httpclient для реализации клиентской части.

Система дизайна тестов используется для создания новых тестов и редактирования существующих. Её легко использовать при минимальном обучении и без навыков программирования. Созданные тестовые данные можно хранить в файлах. Простым решением является использование существующего инструмента, такого как редактор JSON, в качестве системы проектирования тестов [19]. JSON – это формат открытого стандарта, в котором для передачи объектов данных, состоящих из пар атрибут-значение, который способен воспринять человек. Это наиболее распространенный независимый от языка формат данных, используемый для асинхронной связи между браузером и сервером. JSON предлагает ряд преимуществ по сравнению с XML: его можно быстрее анализировать, он более компактен, с ним легче работать на некоторых языках и при форматировании его, как правило, легче читать.

Система контроля тестирования используется для контроля выполнения теста и проверки результатов тестирования. Она имеет следующие возможности: запуск выполнения теста вручную; запускать выполнение теста автоматически после определенного события; автоматический запуск выполнения теста в указанное время; остановка выполнения теста; мониторинг выполнения теста во время выполнения тестов; просмотр журналов тестирования во время выполнения тестов и после них; просмотр отчета тестирования; изменение уровня ведения журнала; выбор регрессионных тестов; настройка тестов. Решение для мониторинга тестирования использует веб-интерфейс для запуска и остановки выполнения теста, а также для создания журналов и отчетов в формате HTML и JSON.

Система выполнения тестов это ядро фреймворка. Его основными компонентами являются скрипты драйверов, тестовая библиотека, анализатор тестовых данных и другие утилиты, такие как генератор журналов и отчетов. Выполнение теста контролируется скриптами драйвера, которые запускаются с помощью системы мониторинга тестирования. Скрипты драйверов имеют несложную структуру, поскольку они в основном используют службы, предоставляемые тестовыми библиотеками и другими повторно используемыми модулями. Библиотеки тестирования содержат функции и модули, которые используются при тестировании или в поддерживающих его действиях. Тестирование – это в основном взаимодействие с тестируемой системой и проверка ее правильности. Функции в тестовой библиотеке могут выполнять свои задачи независимо, но они также могут использовать другие функции или даже внешние инструменты. Например, фреймворк TN3270 используется для реализации работы с z/OS и базой данных IMS, framework Selenium [20] и REST API – веб-интерфейс системы аудита для базы данных IMS. Анализатор тестовых данных предназначен для простого предоставления тестовых данных сценариям драйвера с помощью анализатора JSON в Java. Его задача – обработать тестовые данные и вернуть их скрипту драйвера в простых в использовании контейнерах тестовых данных.

Используемая в нашем исследовании среда автоматизации тестирования была построена с использованием нескольких по-

вторно используемых модулей и библиотек функций, которые имеют следующие особенности:

• Удобство обслуживания: фреймворк значительно снижает затраты на обслуживание.

• Возможность повторного использования: тестовые примеры и функции библиотеки можно использовать повторно.

• Управляемость: эффективный дизайн тестов, отслеживаемость и выполнение.

• Удобство разработки: легко разрабатывать, проектировать, изменять и отлаживать тесты во время выполнения.

• Доступность: позволяет запланировать выполнение автоматизации.

• Надежность: расширенная обработка ошибок и восстановление сценариев.

• Гибкость: структура, независимая от тестируемой системы.

• Измеримость: настраиваемая отчетность о результатах тестирования обеспечивает качественный результат.

Требования высокого уровня к крупномасштабным средам автоматизации тестирования – автоматическое выполнение тестов, простота использования и ремонтопригодность. Наша структура удовлетворяет следующим требованиям [16, 15]:

Требования высокого уровня. Фреймворк автоматически выполняет тесты, прост в использовании без навыков программирования, прост в поддержке, не требует постоянного контроля при выполнении тестов. Это позволяет запускать тесты вручную или планировать автоматическое выполнение в заранее определенное время или после определенных событий. Нефатальные ошибки, вызванные тестовой средой, обрабатываются и сообщаются без остановки выполнения теста. Результаты тестов адекватны и могут быть в ручную перепроверены. Каждый выполненный тест помечается как пройденный или неудачный, а неудачные тесты имеют короткое сообщение об ошибке. Выполнение теста регистрируется с использованием различных настраиваемых уровней ведения журнала. Отчет тестирования создается и публикуется автоматически.

Простота использования. Фреймворк использует подход, основанный на ключевых словах, поддерживает создание пользовательских ключевых слов. Он предоставляет функциональные возможности, которые выбирают и группируют тесты для конкретной задачи.

Удобство обслуживания. Фреймворк является модульным и имеет соглашения о кодировании и именовании. Он реализован с использованием языков сценариев высокого уровня. Тестовое ПО в структуре находится под контролем версий (Git) и должным образом документировано.

Для регрессионного тестирования разработанная автоматизация тестирования требует 3 недели вместо 3 месяцев ручного тестирования.

4. Предлагаемое регрессионное тестирование с полуавтоматическим выбором тестов

Широко используемый метод ускорения регрессионного тестирования называется отбором регрессионного теста (RTS) [1]. RTS сокращает время тестирования за счет повторного запуска только тех тестов, на которые повлияло изменение кода. RTS – надежный метод, если он проверяет все сценарии, на которые может повлиять изменение кода. Если какие-либо сценарии, на которые повлияло изменение кода, не тестируются, регрессией могут быть пропущены. Существующие подходы RTS можно разделить на динамические и статические методы.

Отсутствие практичных инструментов RTS оставляет разработчикам два варианта: они должны либо автоматически перезапустить все тесты, либо выполнить ручной выбор тестов (ручной RTS). Автоматический повторный запуск всех тестов – безопасный подход, но он может быть неточным и неэффективным. Напротив, ручной RTS может быть небезопасным и неточным по нескольким причинам. При использовании RTS вручную разработчики могут выбрать слишком мало тестов и, как следствие, исключить тесты, которые могут выявить различия из-за изменений кода. Они также могут выбрать слишком много тестов и, таким образом, безрезультатно потратить время. Таким образом, разработчики могут извлечь выгоду из автоматизированного метода RTS, который чаще всего работает на практике. Не установлено, насколько ручные методы RTS, используемые разработчиками, сравниваются с автоматизированными методами RTS, предлагаемыми в литературе.

Существует следующие методы выбора регрессионного теста [21]:

• Метод повторного тестирования: этот метод тестирует все тесты, которые ранее выполнялись. Данный метод очень дорогостоя-

щий, так как наборы регрессионных тестов достаточно большие.

• Техника случайного/специального назначения. При использовании этой техники тестировщики полагаются на свой предыдущий опыт и знания, чтобы выбрать, какие тесты необходимо повторить. Это может включать случайный выбор процента покрытия тестов.

• Метод потока данных. Этот метод представляет собой метод выбора регрессионного теста на основе покрытия только тех тестов, которые проверяют взаимодействие данных, имеющих влияние из-за изменений кода.

• Безопасный метод. Этот метод исключает тесты, которые вряд ли обнаружат ошибки. Этот метод обычно выбирает почти все тесты, когда в коде были внесены существенные изменения. Этот метод не фокусируется на критериях покрытия, а выбирает все тесты, которые проверяют результат работы измененной программы по сравнению с ее исходной версией.

• Гибридный метод. Этот метод включает в себя приоритизацию тестовых примеров и минимизацию регрессионных тестов.

В отличие от предыдущих работ [8], мы предлагаем комбинированный подход к выбору последовательности тестов, который использует простые критерии и правила, такие как общие усилия, время выполнения, время развертывания и т.д. Для этого требуются ручные и автоматические настройки.

Основные критерии и правила выбора полуавтоматического регрессионного теста с точки зрения нашего подхода включают:

1. Суммарные усилия, количество необходимых тестов.

2. Требуемое время для тестов, общее время выполнения.

3. Требуемое количество конкретных тестов.

4. Наличие необходимых баз данных IMS, подсистем z/OS.

5. Приоритет тестов.

6. Результаты предыдущих тестов.

Предлагаемое нами регрессионное тестирование с полуавтоматическим выбором тестов отдает приоритет тестам на основе наивысшего приоритета в соответствии с вышеуказанными основными критериями и правилами.

Рассмотрим набор тестов *T*, содержащий *n* тестов, $\{t_1, t_2, \dots, t_n\}$. Для теста t_i определим набор атрибутов $a_i \subseteq A$. Представим как набор функций аудита базы данных IMS, задаваемый вручную тестером. Функциональные возможности аудита базы данных IMS определяются 10 типами фильтрации и 7 типами информации для аудита базы данных IMS.

Для любого t_i теста пусть

$$p_i = \prod_{j=1}^{k} t_i^j$$

обозначает приоритет использования теста, где t_i^j - оценка фактора использования теста на основе j критерия.

Были использованы следующие критерии j для t_i теста в предлагаемом регрессионном тестировании с полуавтоматическим выбором тестов:

• $t^1 = \{0,1\}$ - коэффициент наличия необходимой подсистемы z/OS, версия системы аудита для базы данных IMS, версия базы данных IMS. если доступен и если недоступен.

• $t^2 = [1,10]$ - приоритет теста. Тестер устанавливает этот коэффициент для каждого теста t_i. Например, для высокого приоритета $t^2 = 10$.

• $t^3 = [1, d]$ - коэффициент результата предыдущих тестов. Этот коэффициент рассчитывается для всех предыдущих тестов. d - количество программных сбоев, выявленных тестом t_i.

• $t^4 = [1,20]$ - необходимое количество запусков теста. Тестер устанавливает этот коэффициент для каждого t_i теста. Например, для работ с высоким риском отказа $t^4 = 10$.

• $t^5 = [0,10]$ - коэффициент требований спецификации на основе информации от разработчиков.

• $t^6 = [1,10]$ - коэффициент времени выполнения тестов. Например, для длинного теста $t^6 = 1$ и короткого теста $t^6 = 10$.

Нами был предложен следующий алгоритм, который определяет приоритетность использования тестов для регрессионного тестирования с полуавтоматическим выбором тестов:

1. Определить подсистему z/OS, версию системы аудита для базы данных IMS на z/OS, версию баз данных IMS.

2. Определить критерии t_i, некоторые из них нужно указать вручную.

3. Определить *T'* подмножество тестов в соответствии с подмножеством *A*.

4. Оцените приоритет $\{p_1, \dots, p_n\}$ использования тестов, оценив все критерии.

5. Упорядочить t_i тесты в подмножестве T' согласно значениям p_i.

Для регрессионного тестирования на основе предложенного подхода требуется один день для дымового тестирования и одна неделя для подробного регрессионного тестирования, чтобы удовлетворить требованиям разработчиков и клиентов. Вместо того, чтобы сосредоточиться только на сокращении количества выбранных тестов, наш подход обеспечивает значительное сокращение времени регрессионного тестирования и уравновешивает время, необходимое для анализа, сбора и выполнения тестов.

5. Заключение

Регрессионное тестирование важно для проверки того, что изменения программного обеспечения не нарушают ранее работающую функциональность. Однако регрессионное тестирование обходится дорого, потому, что оно запускает множество тестов для многих версий. Хотя RTS, предложенная более трех десятилетий назад, обеспечивает многообещающий подход к ускорению регрессионного тестирования, ни один метод RTS не получил широкого распространения на практике из-за проблем с эффективностью и безопасностью.

Большинство разработчиков выполняют RTS вручную. Они выбирают тесты специальными способами, которые потенциально не выявляют ошибок или теряя время [4]. Представленный здесь подход предлагает регрессионное тестирование с полуавтоматическим выбором тестов. Этот подход определяет приоритеты тестов на основе приоритета в соответствии с некоторым критерием.

Подход к регрессионному тестированию, который мы представляем здесь, отличается от других подходов тем, что он использует полуавтоматический процесс выбора тестов для определения приоритетности их выполнения. Приоритизация основана на различных критериях, таких как время выполнения, необходимое количество запусков тестов, ручной приоритет тестов, результат предыдущих тестов, функциональность тестов. Этот подход позволяет более детально настроить последовательность регрессионных тестов вручную.

Представленный в данной работе подход обеспечивает существенное сокращение времени, необходимого для регрессионного тестирования, и уравновешивает время, необходимое для анализа, сбора и выполнения, вместо того, чтобы сосредоточиться исключительно на сокращении количества выбранных тестов.

В данном исследовании оценивалась осуществимость представленной среды автоматизации тестирования для регрессионного тестирования системы аудита баз данных IMS. Общая оценка была положительной, и фреймворк был признан эффективным. Представленная структура автоматизации тестирования удовлетворяет важным требованиям более низкого уровня с точки зрения простоты использования и удобство обслуживания.

Литература

1. Legunsen O., Hariri F., Shi A., Lu Y., Zhang L., Marinov D. An extensive study of static regression test selection in modern software evolution. Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2016, pp. 583–594.

2. Tulpule N. Strategies for testing client-server interactions in mobile applications: How to move fast and not break things. Proceedings of the 2013 ACM Workshop on Mobile Development Lifecycle, 2013, pp. 19–20.

3. Jensen C. S., Moller A., Su Z. Server interface descriptions for automated testing of javascript web applications. Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, 2013, pp. 510–520.

4. Gligoric M., Eloussi L., Marinov D. Practical regression test selection with dynamic file dependencies. Proceedings of the 2015 International Symposium on Software Testing and Analysis, 2015, pp. 211–222.

5. Rothermel G., Harrold M. J. A safe, efficient regression test selection technique. ACM Trans. Softw. Eng. Methodol., 1997, vol. 6, no. 2, pp. 173–210.

6. Shi A., Yung T., Gyori A., Marinov D. Comparing and combining test-suite reduction and regression test selection. Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, 2015, pp. 237–247.

7. Zhong H., Zhang L., Mei H. An experimental study of four typical test suite reduction techniques. Information and Software Technology, 2008, vol. 50, no. 6, pp. 534–546.

8. Hao D., Zhang L., Zhang L., Rothermel G., Mei H. A unified test case prioritization approach. ACM Trans. Softw. Eng. Methodol., 2014, vol. 24, no. 2, pp. 10:1–31.

9. Zhai K., Jiang B., Chan W. K. Prioritizing test cases for regression testing of location-based services: Metrics, techniques, and case study. IEEE Trans. Serv. Comput., 2014, vol. 7, no. 1, pp. 54–67.

10. Yoo S., Harman M. Regression testing minimization, selection and prioritization: A survey. Softw. Test. Verif. Reliab., 2012, vol. 22, no. 2, pp. 67–120.

11. Klein B., Long R. A., Blackman K. R., Goff D. L., Nathan S. P., Lanyi M. M., Wilson M. M., Butterweck J., Sherrill S. L. An Introduction to IMS: Your Complete Guide to IBM Information Management System. IBM Press, 2nd ed., 2012.

12. Burleson D. K. Inside the Database Object Model. Boca Raton, FL, USA: CRC Press, Inc., 1st ed., 1998.

13. Long R., Harrington M., Hain R., Nicholls G. IMS Primer. USA: IBM Redbook, 2000.

14. Bajpai R. N. A Keyword Driven Framework for Testing Web Applications. International Journal of Advanced Computer Science and Applications (IJACSA), 2012, vol. 3, no. 3.

15. Abhishek K., Kumar P., Sharad T. Automation of Regression Analysis: Methodology and Approach, 2012, pp. 481–487.

16. Kumar P., Kavita D. Automation framework for database testing. 5th International Conference on Recent Innovations in Science, Engineering and Management, 2016, pp. 88–93.

17. Laukkanen P. Data-driven and keyword-driven test automation frameworks, 2007.

18. Artzi S., Dolby J., Jensen S. H., Moller A., Tip F. A framework for automated testing of javascript web applications. Proceedings of the 33rd International Conference on Software Engineering, 2011, pp. 571–580.

19. Sabev P., Grigorova K. Manual to automated testing: An effort-based approach for determining the priority of software test automation. International Journal of Computer, Electrical, Automation, Control and Information Engineering, 2015, vol. 9, no. 12, pp. 2456–2462.

20. Gojare S., Joshi R., Gaigaware D. Analysis and design of selenium webdriver automation testing framework. Procedia Computer Science, 2015, vol. 50, pp. 341–346.

21. Jyoti K. S. A comparative study of five regression testing techniques: A survey. International journal of scientific & technology research, 2014, vol. 3, pp. 76–80.

22. Ruchay A.N. A regression testing with semi-automatic test selection for auditing of ims database // Chelyabinsk Physical and Mathematical Journal, 4(2), 2019. Pp. 241-249.

23. Агафонов А.В., Синадский Н.И. Тестирование защищенности телекоммуникационного оборудования от сетевых компьютерных атак типа «отказ в обслуживании» с применением генетического алгоритма // Вестник УрФО. Безопасность в информационной сфере № 2(24), 2017. С. 4-8.

References

1. Legunsen O., Hariri F., Shi A., Lu Y., Zhang L., Marinov D. An extensive study of static regression test selection in modern software evolution. Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2016, pp. 583–594.

2. Tulpule N. Strategies for testing client-server interactions in mobile applications: How to move fast and not break things. Proceedings of the 2013 ACM Workshop on Mobile Development Lifecycle, 2013, pp. 19–20.

3. Jensen C. S., Moller A., Su Z. Server interface descriptions for automated testing of javascript web applications. Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, 2013, pp. 510–520.

4. Gligoric M., Eloussi L., Marinov D. Practical regression test selection with dynamic file dependencies. Proceedings of the 2015 International Symposium on Software Testing and Analysis, 2015, pp. 211–222.

5. Rothermel G., Harrold M. J. A safe, efficient regression test selection technique. ACM Trans. Softw. Eng. Methodol., 1997, vol. 6, no. 2, pp. 173–210.

6. Shi A., Yung T., Gyori A., Marinov D. Comparing and combining test-suite reduction and regression test selection. Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, 2015, pp. 237–247.

7. Zhong H., Zhang L., Mei H. An experimental study of four typical test suite reduction techniques. Information and Software Technology, 2008, vol. 50, no. 6, pp. 534–546.

8. Hao D., Zhang L., Zhang L., Rothermel G., Mei H. A unified test case prioritization approach. ACM Trans. Softw. Eng. Methodol., 2014, vol. 24, no. 2, pp. 10:1–31.

9. Zhai K., Jiang B., Chan W. K. Prioritizing test cases for regression testing of location-based services: Metrics, techniques, and case study. IEEE Trans. Serv. Comput., 2014, vol. 7, no. 1, pp. 54–67.

10. Yoo S., Harman M. Regression testing minimization, selection and prioritization: A survey. Softw. Test. Verif. Reliab., 2012, vol. 22, no. 2, pp. 67–120.

11. Klein B., Long R. A., Blackman K. R., Goff D. L., Nathan S. P., Lanyi M. M., Wilson M. M., Butterweck J., Sherrill S. L. An Introduction to IMS: Your Complete Guide to IBM Information Management System. IBM Press, 2nd ed., 2012.

12. Burleson D. K. Inside the Database Object Model. Boca Raton, FL, USA: CRC Press, Inc., 1st ed., 1998.

13. Long R., Harrington M., Hain R., Nicholls G. IMS Primer. USA: IBM Redbook, 2000.

14. Bajpai R. N. A Keyword Driven Framework for Testing Web Applications. International Journal of Advanced Computer Science and Applications (IJACSA), 2012, vol. 3, no. 3.

15. Abhishek K., Kumar P., Sharad T. Automation of Regression Analysis: Methodology and Approach, 2012, pp. 481–487.

16. Kumar P., Kavita D. Automation framework for database testing. 5th International Conference on Recent Innovations in Science, Engineering and Management, 2016, pp. 88–93.

17. Laukkanen P. Data-driven and keyword-driven test automation frameworks, 2007.

18. Artzi S., Dolby J., Jensen S. H., Moller A., Tip F. A framework for automated testing of javascript web applications. Proceedings of the 33rd International Conference on Software Engineering, 2011, pp. 571–580.

19. Sabev P., Grigorova K. Manual to automated testing: An effort-based approach for determining the priority of software test automation. International Journal of Computer, Electrical, Automation, Control and Information Engineering, 2015, vol. 9, no. 12, pp. 2456–2462.

20. Gojare S., Joshi R., Gaigaware D. Analysis and design of selenium webdriver automation testing framework. Procedia Computer Science, 2015, vol. 50, pp. 341–346.

21. Jyoti K. S. A comparative study of five regression testing techniques: A survey. International journal of scientific & technology research, 2014, vol. 3, pp. 76–80.

22. Ruchay A.N. A regression testing with semi-automatic test selection for auditing of ims database // Chelyabinsk Physical and Mathematical Journal, 4(2), 2019. Pp. 241-249.

23. Agafonov A.V.. Sinadskiy N.I. Testirovaniye zashchishchennosti telekommunikatsionnogo oborudovaniya ot setevykh kompyuternykh atak tipa «otkaz v obsluzhivanii» s primeneniyem geneticheskogo algoritma // Vestnik UrFO. Bezopasnost v informatsionnoy sfere № 2(24). 2017. C. 4-8.

РУЧАЙ Алексей Николаевич, кандидат физико-математических наук, доцент, заведующий кафедрой компьютерной безопасности и прикладной алгебры, Челябинский государственный университет. 454001, Россия, г. Челябинск, ул. Братьев Кашириных, 129; доцент кафедры защиты информации, Южно-Уральский государственный университет (национальный исследовательский университет). 454080, Россия, г. Челябинск, пр. им. В.И. Ленина, 76. E-mail: ran@csu.ru

RUCHAY Alexey, PhD in Physics and Mathematics, Associate Professor, Head of the Department of Computer Security and Applied Algebra, Chelyabinsk State University. 454001, Russia, Chelyabinsk, st. Kashirin Brothers, 129; Associate Professor, Department of Information Security, South Ural State University (National Research University), 454080, Russia, Chelyabinsk, etc. them. IN AND. Lenin, 76. E-mail: ran@csu.ru