зміни; $z_з$ - параметр, якій відповідає за кількість поїздів, які надходять на станцію при заборонному сигналі світлофора, протягом зміни; $z_з^{заб}$ - параметр, якій відповідає за кількість поїздів, які відправляються зі станції при заборонному показанні вихідного світлофора (в наслідку реалізації даної дії поїзд може бути відправлений на фактично зайнятий перегін), протягом зміни.

Висновки. Запропонований параметр $K_б$ в такому вигляді, як його було визначено у виразах (1) і (2) не дозволяє адекватно оцінювати безпеку при виконанні експлуатаційної роботи. Відповідно до цього в подальших дослідженнях постає задача трансформування параметру $K_б$ в такий вигляд, який дозволить найбільш об'єктивно оцінювати ступень безпеки з урахуванням перелічених факторів згідно наведених раніше параметрів.

Оскільки задача формування параметру безпечної експлуатації $K_б$ слабо структурована по своїй сутності і вирішення її буде ґрунтуватися на узагальненні ряду незалежних та різних, а в деяких випадках і зовсім безрозмірних параметрів та факторів. Відповідно до цього найбільш доцільним в даному випадку є застосування теорії нечіткої логіки, яка дозволяє адекватно вирішувати задачі таких класів [4].

Вирішення задачі трансформації параметру безпечної експлуатації в подальшому доцільно вирішувати на основі формування ряду функцій належності, призначенням яких є відтворення та врахування набору визначених у виразі (2) факторів та параметрів.

Сформований параметр безпечної експлуатації може бути використаним у якості основного важеля при обґрунтуванні доцільності обладнання робочих місць оперативного персоналу автоматизованими засобами.

Список літератури: **1**. Пособие поездному диспетчеру и дежурному по отделению [Текст] – М.: Транспорт, 1992. – 369с**. 2**. *Кочнев Ф. П.* Управление эксплуатационной работой железных дорог [Текст] / *Ф. П. Кочнев, И. Б. Сотников* – М.: Транспорт, 1990. – 424 с. **3**. *Лаврухін О. В.* Формування критерію безпеки для оцінки транспортної події – прийняття поїзда на зайняту колію [Текст] / *О. В. Лаврухін* // Інформаційно-керуючі системи на залізничному транспорті. Науково-технічний журнал. – Х.:, 2011. – Вип. 2. – С. 102-108. **4.** *Мелехов А. Н.* Ситуационные советующие системы с нечёткой логикой [Текст] / *А. Н. Мелехов, Л. С. Бернштейн, С. Я. Коровин* – М.: Наука. Гл. ред. Физ.-мат.-лит. – 1990.

Надійшла до редколегії 20.03.2013

УДК 656.222.3:658.5

Формування параметру безпечного управління поїзною роботою на залізничній станції/ Лаврухін О. В. // Вісник НТУ «ХПІ». Серія: Нові рішення в сучасних технологіях. – Х: НТУ «ХПІ», – 2013. - № 68 (978). – С. 30-33. – Бібліогр.: 4 назв.

Определены основные критерии, которые влияют на безопасность движения и сформирован параметр безопасного управления поездной работой на железнодорожных станциях.

Ключевые слова: безопасность движения, человеческий фактор, параметр безопасного управления, опасный груз, функция принадлежности.

The main criteria that affect traffic safety and shaped control security has been showed at railway stations.

Keywords: traffic safety, human factors, control security, dangerous goods, the membership function.

УДК 004.358:681.518

А. В. НИКИТИНА, студент, ХНУРЭ, Харьков

СИСТЕМА АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ СЕРВЕРНОЙ ЧАСТИ WEB-ПРИЛОЖЕНИЙ

Ми представляємо систему автоматизованого тестування серверної частини web-додатків, яка дозволить розробляти тестові сценарії для функціонального тестування одночасно зі створення необхідної документації - тест-дизайну.

Ключові слова: автоматизоване тестування; функціональне тестування; модель тестування; тест-кейс.

Вступ. Автоматизація тестування програмного забезпечення означає створення спеціальних скриптів для емуляції роботи кінцевого користувача з системою. Спеціалізована програма буде виконувати заздалегідь певний набір дій, відправляти запити БД, перевіряти і аналізувати результат тесту.

Автоматизоване тестування надає ряд переваг, якщо воно вірно впроваджено і точно відповідає процесу. Переваги автоматизованого тестування перед ручним очевидні - це і висока швидкість виконання тестів, і можливість виконувати однотипні тести знову і знову. Воно також допомагає виявити помилки, які були пропущені на стадії ручного тестування. Автоматизація тестів забезпечує підтримку Agile і екстремальних методів програмування, дозволяє підтримувати сувору документацію проекту і швидко надавати звіт про стан продукту. У позитивних сторонах автоматизації, на щастя, ніхто вже не сумнівається. Однак, не дивлячись на це, дуже рідко вдається перейти повністю від ручного тестування до автотестування.

Автоматизувати тестування має сенс у довгострокових проектах, де постійно існує можливість того, що певний функціонал перестане працювати як необхідно із введенням нових функцій.

Для таких проектів також важливо вести постійну документацію. Одним з основних видів документації з тестування є Test Case.

Test Case - це артефакт, що описує сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації тестованої функції або її частини.

Так як для якісного тестування, особливо регресійного, важливим являється створення Test Case, за якими з рештою буде виконуватися автоматичне тестування, то було вирішено створити додаток у якому можна було б розробляти тестові сценарії за певним шаблоном, за створеним тестовим сценарієм формувати клас для виконання цього сценарію із пустими методами, у яких тестувальник буде реалізовувати функціонал для проходження певного кроку. А також реалізація можливості проходження створеним тестовим засобом по кроках, що описані у Test Case.

Аналіз моделей тестування. Імовірнісне тестування.

Ця модель передбачає, що відомий розподіл наступних випадкових величин (можливо, тільки першої або двох перших)

– частота використання різних впливів на цільове ПО;

– втрати від виникнення помилки при даному впливі;

– ресурси, необхідні для виправлення помилки при даному впливі.

На їх основі набір тестових впливів генерується випадковим чином так, щоб їх розподіл мінімізував витрати на виправлення, втрати або просто ймовірність виникнення помилки.

Приклад. Нормально розподілені double в якості аргументу функції sin.

Комбінаторне тестування.

Ця модель припускає завдання деякої структуризації на вхідних впливах, при якій одному впливу відповідає деяка комбінація атомарних елементів.

Генерація тестів грунтується на процедурі систематичного перебору комбінацій такого роду.

Метод функціональних діаграм дає одну з процедур такого роду.

Приклад. Операція додавання цілих чисел і дає ціле число. Цілі числа бувають негативні, позитивні і 0. В якості набору тестових впливів можна запропонувати всі комбінації варіантів для аргументів і результату.

Тестування на основі автоматів.

Тестові впливи генеруються як послідовність впливів, що призводить до проходу по

деякому маршрутом в автоматній моделі.

Приклад зображено на рис. 5.

Для наведеного автомата підходящої послідовністю тестових впливів буде aaababaabb.

Вона дає обхід цього автомата - маршрут в ньому, що містить всі переходи.

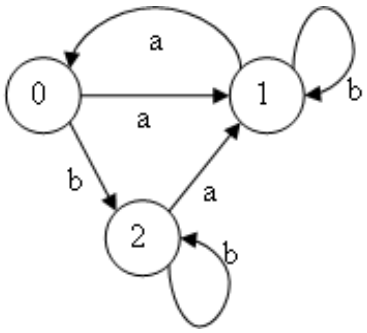

Рис. 1 - Прохід по маршруту в автоматній моделі

Тестування з алгебраїчним моделям.

Використовується набір аксіом у вигляді еквівалентності ланцюжків операцій. Береться ланцюжок для «затравки», а додаткові послідовності тестів будуються як результати її перетворення за заданими правилами еквівалентності. Тест на основі аналізу трас виконання перевіряє, що всі такі послідовності приводять до одного результату.

Приклад. Stack

pop (); push (a); == return a;

pop (); {pop (); push (b);} push (a) == {return b;} return a; і т.д.

Перші два види моделей застосовні тільки для систем, поведінка яких не залежить від історії[4].

Для першої моделі це можна поправити, використовуючи замість звичайних розподілів випадкові процеси. Зазвичай використовуються марківські ланцюги.

Перша модель вимагає великих знань про шаблони використання ПЗ і про витрати на можливі помилки (для марковських ланцюгів кількість необхідних знань зростає).

Остання модель не дає обгрунтованої відповіді на питання: які затравочні послідовності використовувати, а які ні, і скільки перетворень достатньо для досягнення потрібного покриття.

Друга модель використовується для генерації тестових впливів в одному стані.

Тому для створення системи автоматизованого тестування серверної частини web-додатків найбільш підходить третя модель.

Постановка задачі. Для тестування серверної частини веб-додатків зручно використовувати FitNesse - це інструмент тестування програмного забезпечення. У ньому можна написати тестові сценарії на wiki-сторінці у вигляді таблиць із назвами функцій і вхідними та вихідними даними і зв'язати їх із програмним кодом, який буде виконуватись відповідно до кроків описаних у тесті.

Дана концепція є зручною для тестування серверної частини веб-додатків. Тому було вирішено розробити систему, яка була би своєрідним продовженням даної технології. А саме розробити додаток, який би замість wiki-сторінок використовув шаблони тест-кейсів, а також створював за цими тест-кейсами клас з методами, у яких уже будуть задані вхідні значення. Тестувальнику необхідно буде лише написати реалізацію для кожного тестувального методу.

Система має працювати із серверною частиною веб-додатків і виконувати автоматизоване функціональне тестування за написаними у самому додатку тест-кейсами.

Для цього необхідно реалізувати шаблон за яким будуть створюватися тест-кейси. Створити базу даних, у якій будуть зберігатися написані тест-кейси, а також результат їх виконання.

Необхідно розробити функціонал для конвертації тест-кейсів у шаблон програмного коду, у якому буде створено клас за назвою тесту, а також методи із вхідними параметрами, кожен метод має відповідати певному кроку описаному у тест-кейсі.

Для можливості проходження заповненого шаблону програмного коду по крокам описаним у тест-кейсі необхідно реалізувати відповідний зв'язок між цими компонентами, а також між самим веб-додатком.

Вхідними даними програмної системи є описані кроки у шаблоні тест-кейсу і програмна реалізація методів, що відповідають цим крокам.

Вихідними даними програмної системи є результат виконання тестів.

Опис роботи системи. Приклад заповненого шаблону тест-кейсу зображено на рис. 2.

ID **Author** **Name** **Variables:** [Login]:string= "User" [Pass]:string="Pass" [ActualMessages]:int [InactualMessages]:int			
Steps	Expected result	Actual result	Pass/Fail
Input login and pass {methodName([Login], [Pass])}	True	methodName	
Check the count of active messages for this account{countActiveMessages([Login]:string, [Pass] :string):int}	12	[ActualMessages] = countActiveMessages	
Check the count of inactive messages{countOfInactiveMessages([Login], [Pass]):int }		[InactualMessages] = countOfInactiveMessages	
Equals active messages with inactive {DifferentBetweenActiveAndInactive([ActualMessages], [InactualMessages])}	0	DifferentBetweenActiveAndInactive	
{Log off()}			

Рис. 2 - Приклад заповненого шаблону тест-кейсу.

Пояснення до шаблону:

{} - ідентифікатор виклику методу;

[] - ідентифікатор змінних всередині TestCase;

"" - ідентифікатор значення змінних;

: - після цього знаку вказується тип, який буде повертати метод або тип переданих змінних.

Якщо метод написаний з () то значить, що його треба викликати з параметрами і створити такий в класі. Якщо написано тільки назва методу, то значить потрібно тільки те значення, яке цей метод повертає, тому в колонці Actual result пишеться тільки його назва, тому що там потрібен тільки його результат. Результат виконання методу можна записати в змінну і потім передавати її в метод.

Також можна просто викликати метод, який не приймає параметрів і не повертає нічого, тоді поля Expected result, Actual result для нього не заповнюються.

Після заповнення шаблону TestCase необхідно натиснути на кнопку "Form class" і тоді під цей шаблон сформується клас, називатися він буде також як і тест-кейс, в якому будуть тільки ті методи які викликаються в тест-кейсы. Це будуть методи-пустушки, так як вони будуть мати такий вигляд:

public returntype NameOfClass (parameters)

{}

Далі користувач вже сам заповнює вміст кожного методу. А також прописує шлях до класу, щоб шаблон із ним міг зв'язатися.

Після того як всі методи заповнені клас обробки повинен сам зв'язатися з тест-кейсом, у якого буде таке ж ім'я як і у класу і виконувати послідовно всі написані кроки (методи). А в повернутому тест-кейсі повинні відображатися замість назви змінних їх значення, а також в полі Pass / Fail написаний результат порівняння очікуваного і поточного результату. Якщо хоча б один із кроків має результат Fail тест позначається як Failed і його ідентифікатор розміщується в таблицю бази даних "FailedTests"

Висновки. У рамках проведеного дослідження існуючої проблематики та аналізу існуючих моделей автоматизованого тестування web-додатків було прийнято рішення про доцільність використання моделі тестування на основі автоматів, для створення системи і проектування її поведінки.

Список літератури: **1.** *Дастін Е.* Автоматизоване тестування програмного забезпечення [Текст] / *Е. Дастін, Д. Решка, Д. Пол* - М.:ЛОРИ, 2003.-590с. **2.** *Канер С.* Тестування програмного забезпечення [Текст] / *С. Канер, Д. Фолк, Е. Кек Нгуен* - М.: ДиаСофт, 2001.-538с. **3**. *Майерс Г. Д.* Надійність програмного забезпечення [Текст] / *Г.Д. Маейрс*. - М.: Мир, 1980. - 355с. **4.** *Бейзер Б.* Тесутвання чорної скриньки. Технології функціонального тестування програмного забезпечення і систем [Текст] / *Б. Бейзер*. - С.: Питер, 2004. - 320с. **5**. *Тамре Л.* Тестування програмного забезпечення [Текст] / *Л. Тамре*. - М.:Вильямс, 2003. - 354с. **6**. *Бек К.* Екстремальне програмування. Розробка через тестування [Текст] / *К. Бек*. - С.: Питер, 2003. - 224с. **7**. *Капбертсон Р.* Швидке тестування [Текст] / *Р. Капбертсон, К. Браун, Г. Кобб*. - М.: Вильямс, 2002.-384с. **8**. *Блек Р.* Ключові процеси тестування [Текст] /*Р. Блек*. - М.:ЛОРИ, 2011.-544с. **9**. *Майерс Г. Д.* Мистецтво тестування програм [Текст] / *Г. Д. Маейрс*. - М.: Финансы и статистика, 1982. - 272с. **10**. *Савін Р.* Тестування Дот Ком. [Текст] / *Р. Савін*. - М.:Дело, 2008.-312с.

Надійшла до редколегії 20.03.2013

УДК 004.358:681.518

Система автоматизированного тестирования серверной части WEB-приложений/ Никитина А. В. // Вісник НТУ «ХПІ». Серія: Нові рішення в сучасних технологіях. – Х: НТУ «ХПІ», – 2013. - № 1 (977). – С. 33-37. – Бібліогр.: 10назв.

Мы представляем систему автоматизированного тестирования серверной части web-приложений, которая позволит разрабатывать тестовые сценарии для функционального тестирования одновременно с созданием необходимой документации - тест-дизайна.

Ключевые слова: автоматизированное тестирование; функциональное тестирование; модель тестирования; тест-кейс.

We present a system of automated testing of web-server application that allows to develop test scripts for functional testing along with the creation of the necessary documentation - test design.

Keywords: automated testing, functional testing, model testing, test cases.

УДК 656.222.3

Г. Я. МОЗОЛЕВИЧ, канд. техн. наук, доц., ДНУЗТ ім.академіка В.Лазаряна, Дніпропетровьк

ПІДВИЩЕННЯ КОНКУРЕНТОСПРОМОЖНОСТІ ЗАЛІЗНИЧНИХ ПЕРЕВЕЗЕНЬ ШЛЯХОМ УПРАВЛІННЯ ПАРАМЕТРАМИ ВАНТАЖНИХ ПОЇЗДОПОТОКІВ

В статті запропоновано за рахунок управління параметрами маси та довжини поїздів зменшити загальні витрати учасників логістичного ланцюгу вантажопотоків.

Ключові слова: залізничний напрямок, маса та довжина поїздів, конкурентоспроможність залізниць.

Вступ. Сучасні умови функціонування залізничного транспорту України характеризуються постійною зміною структури й обсягів вантажо- та поїздопотоків при наявності резервів пропускної спроможності більшості дільниць залізничних напрямків перевезень. В зв'язку з цим зростає важливість проблеми вибору раціональних параметрів маси та довжини поїздів з метою зменшення експлуатаційних витрат залізниці та витрат клієнтів як єдиної системи.

Залізничний транспорт одночасно працює в ринкових умовах конкуренції з іншими видами транспорту, і в той же час, згідно закону України про транспорт, його головним завданням є своєчасне, повне і якісне задоволення потреб населення та суспільного виробництва в перевезеннях. При цьому залізничний транспорт виступає монополістом в