Web-აპლიკაციების ავტომატიზებული ტესტირება

მიხეილ გულიტაშვილი, გია სურგულაძე
საქართველოს ტექნიკური უნივერსიტეტი

რეზიუმე

განიხილება პროგრამული სისტემების სასიცოცხლო ციკლის მენეჯმენტის საკითხი, კერძოდ მისი ტესტირების ეტაპი. შემოთავაზებულია პროგრამული უზრუნველყოფის ტესტირების სისტემების კლასიფიკაცია, მათ შორის **Web-**აპლიკაციების ტესტირების ავტომატიზაციის საფუძვლები. ილუსტრირებულია ტესტ-სკრიპტების აგების და ფუნქციონირების კონკრეტული მაგალითი. აღიწერება ავტომატური ტესტირების გამოყენების სფეროები თანმედროვე ინფორმაციულ ტექნოლოგიებში.

საკვანძო სიტყვები: პროგრამული ინჟინერია. პროგრამების სასიცოცხლო ციკლი. ვებ-აპლიკაცია. ავტომატური ტესტირება. ტესტ-სკრიპტი.

1. შესავალი

თანამედროვე ინფორმაციულ ტექნოლოგიებში გვხვდება ორი ტიპის Windows- და Web-აპლიკაციები. ორივე ტიპის პროგრამული პაკეტები გამოიყენება ორგანიზაციული მენეჯმენტის პროცესების ავტომატიზაციისთვის [1]. კერძოდ გამოთვლების სისწრაფისა და საიმედოობის ასამაღლებლად, შეცდომების თავიდან ასაცილებლად და ა.შ. დღეისათვის მსოფლიოში Web აპლიკაციების განვითარების ბუმია, რადგან ასეთი ტიპის გამოყენებით პროგრამებზე წვდომა უფრო მარტივად ხორციელდება ინტერნეტის საშუალებით, სადაც შესაძლებელია ასევე კარგი პროგრამული დაცვის სისტემების აგება, ქსელური შეტევების თავიდან აცილების მიზნით. დღეისთვის გავრცელებული Web პროგრამული სისტემებია: ERP (Enterprise Resource Planning), CRM (Customer Relationship Management), HR (Human Resources), CMS (Content Management System), Social Network და ა.შ. [2-4].

თანამედროვე ინფორმაციული ტექნოლოგიები გვთავაზობს Web-ზე ბაზირებული პროგრამული სისტემების ავტომატურ ტესტირებას, პროგრამული პროდუქტის ტესტირების პროცესის ავტომატიზაციას [1]. დღეისთვის გავრცელებული პროგრამული აპლიკაციების უმრავლესობა არის Web-ზე ბაზირებული, რომელთა მოხმარება შესაძლებელია ინტერნეტ ბრაუზერის მეშვეობით. ასეთი ტიპის პროგრამები ფართოდაა გავრცელებული კომპანიებსა და ორგანიზაციებში, სასწავლო დაწესებულებებში, ბანკებში, სათამაშო ბიზნესში და ა. შ. Web პროგრამების ტესტირება და ხარისხის კონტროლი არის პროგრამული პროდუქტის სასიცოცხლო ციკლის თანმხლები ეტაპი [6]. ტესტირების ავტომატიზაცია ნიშნავს რომ სხვა პროგრამული უზრუნველყოფის საშუალებით გავუშვათ განმეორებადი ტესტები მოცემული აპლიკაციის დასატესტირებლად [4].

ტესტირების ავტომატიზაციისას მრავალი სირთულეა, რომელიც დაკავშირებულია განმეორებადი ტესტების შექმნასთან, ტესტის გაშვების სიჩქარის განსაზღვრასთან, სატესტო მონაცემთა ბაზის შექმნასთან და სხვ. განმეორებადი ტესტის შექმნა გულისხმობს ისეთი მატესტირებელი სცენარის დაწერას რომლის გაშვებაც მრავალჯერადად არის შესაძლებელი. დღეისთვის გავრცელებულია მრავალი კერძო და ღია პროგრამული პროდუქტი, რომლებიც უზრუნველყოფს Web აპლიკაციების ავტომატურ ტესტირებას. ფასიანი პროდუქტებიდან შეიძლება გამოვყოთ QTP (HP QuickTest Professional) [5]. უფასო პროდუქტებიდან: Selenium, Watir, Canoo WebTest, Cucumber, CubicTest და სხვ. [1].

ცხადია თუ მოვახდენთ ჩვენი პროგრამული პროდუქტის ავტომატურ ტესტირებას, ეს იქნება გარანტი შეცდომების თავიდან აცილების, პროგრამის საიმედოობს, ტესტირების დროის დაზოგვის, ტესტირების პროცესის დაჩქარების. მაგრამ არის პროგრამების ავტომატური ტესტირება ყოველთვის ხელსაყრელი? ის არ არის ყოველთვის მიზანშეწონილი, მაგალითად წარმოვიდგინოთ პროგრამული პროდუქტი, რომლის სამომხმარებლო ინტერფეისი ხშირად იცვლება, ამ შემთხვევაში მატესტირებელი სკრიპტების ცვლილება გარდაუვალია. ასევე როცა დასატესტირებელი პროგრამა პატარა და მარტივია, ამ შემთხვევებში ხელით ტესტირება უფრო აპრობირებულია ვიდრე ტესტირების ავტომატიზაცია.

2. ძირითადი ნაწილი

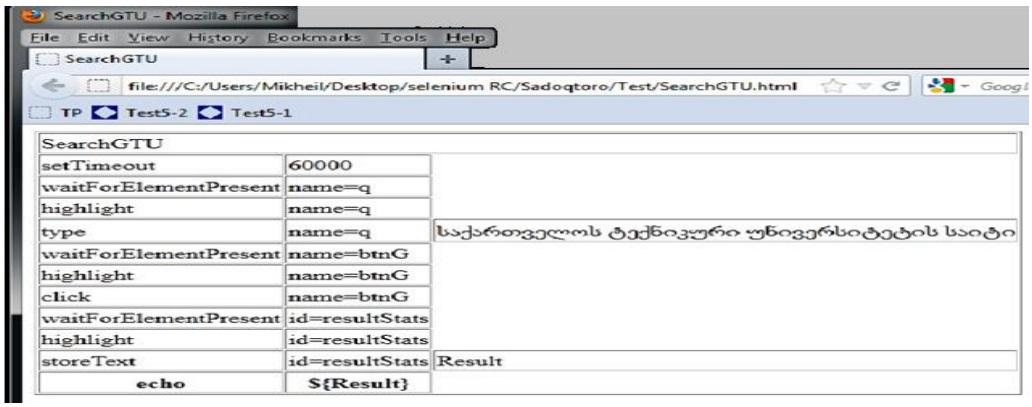
Web პროგრამული სისტემების ავტომატური ტესტირება ფართოდ გამოყენებადი პროცესია თანამედროვე ინფორმაციულ ტექნოლოგიებში. პროგრამული უზრუნველყოფის ტესტირება ერთ-ერთი აუცილებელი და მნიშვნელოვანი ფაზაა პროგრამული პროდუქტის დეველოპმენტისას [6]. ხოლო თუ მოხდება ამ პროცესის ავტომატიზაცია, იგი მოგვცემს საკმაოდ დიდი რესურსების დანაზოგს.

მაგალითისთვის განვიხილოთ პრაქტიკული ამოცანა, რომლის საშუალებითაც აღვწერთ ტესტირების ავტომატიზაციის პროცესს. ამოცანის მიზანია, რომ ,,გუგლის“ საშუალებით მოვახდინოთ ,,საქართველოს ტექნიკური უნივერსიტეტის" საიტის ძებნის ავტომატიზაცია.

ამ ამოცანის გადასაწყვეტად გამოვიყენოთ უფასო პროგრამული უზრუნველყოფა Selenium, რომლის საშუალებითაც შეიძლება ტესტირების ავტომატიზაცია ნებისმიერ ოპერაციულ სისტემაზე და ნებისმიერ ინტერნეტ ბრაუზერზე. იმისთვის რომ მოვახდინოთ ტესტირების ავტომატიზაცია საჭიროა დავწეროთ სცენარი (იხ. ცხრ.1), ეგრეთ წოდებული Test Scenario, რომლის მიხედვითაც მოხდება შემდეგ მოვლენათა თანამიმდევრობის შესრულება: **ცხრ.1**

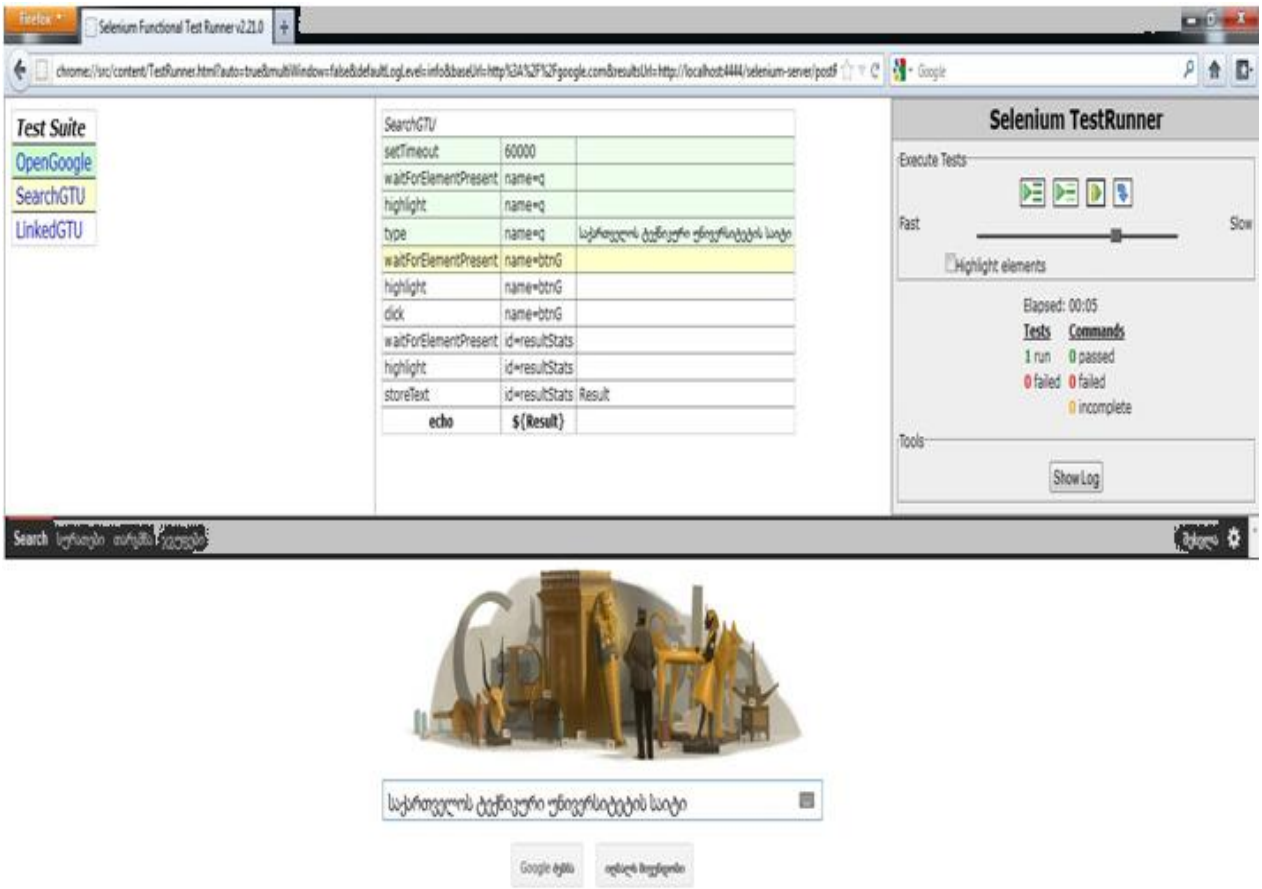
№	მოვლენა
1	გაუშვი Firefox (ან სხვა ბრაუზერი)
2	დაუკავშირდი და გახსენი საძიებო სისტემა Google
3	ჩაწერე ,,საქართველოს ტექნიკური უნივერსიტეტის საიტი"
4	დააჭირე ღილაკს "Google Search"
5	გადადი პირველ მოძებნილ ლინკზე
6	დაიმახსოვრე გახსნილი ვებ-გვერდის მისამართი

აღწერილი სცენარის მიხედვით იწერება შესაბამისი ტესტ-სკრიპტები, რომლთა თანამიმდევრობაც ქმნის TestSuite-ს. 1-ელ ნახაზზე მოცემულია ტესტ-სკრიპტის მაგალითი:



ნახ.1

მე-2 ნახაზზე მოცემულია ავტომატური ტესტირების პროცესი, Selenium-ით Firefox ბრაუზერის გაშვება და ძებნა:



ნახ.2

ტესტირების დასრულების შედეგად გენერირდება ტესტირების რეპორტი, საშედეგო ფაილი, რომლიც გვაძლევს სცენარის შესრულების შედეგის ანალიზის საშუალებას. ჩვენი მაგალითის საშედეგო ფაილი მოცემულია მე-3 ნახაზზე, საიდანაც ჩანს რომ გუგლის მიერ მოძებნილი "საქართველოს ტექნიკური უნივერსიტეტის" საიტის მისამართი ყოფილა http://www.gtu.edu.ge.

მატესტირებელი ტესტ-სკრიპტების დაწერისთვის გამოიყენება სპეციალური ბრძანებები რომლთა დასახელებებს განსაზღვრავს ტესტირების ენის სინტაქსი. Selenium ბრძანებების ერთობლიობას, რომელთა საშუალებით ხდება ტესტების გაშვება ეწოდება Selenese. Selenese ბრძანებების მიმდევრობა არის სწორედ ტესტ-სკრიპტი [4].

ტესტირების სცენარის შესაქმნელად, ტესტ-სკრიპტების დაწერისთვის გამოიყენება პროგრამა Selenium IDE, რომელიც წარმოადგენს Firefox-ის პლაგინს. მისი საშუალებით შესაძლებელია სკრიპტების ავტომატურ რეჟიმში ჩაწერა. გაშვებულ რეჟიმში მყოფი Selenium IDE ავტომატურად ჩაწერს ჩვენი მოქმედების შესაბამის ბრძანებებს [4]. მისი საშუალებით შეიძლება ტესტ-სკრიპტების შექმნის პროცესის ავტომატიზაცია, ეს შეამცირებს სკრიპტების შექმნაზე დახარჯულ დროს.

Test Suite
OpenGoogle
SearchGTU
LinkedGTU

./OpenGoogle.htm

OpenGoogle		
windowMaximize		
setTimeout	60000	
setSpeed	700	
open	http://www.google.com	

./SearchGTU.html

SearchGTU		
setTimeout	60000	
waitForElementPresent	name=q	
highlight	name=q	
type	name=q	საქართველოს ტექნიკური უნივერსიტეტის საიტი
waitForElementPresent	name=btnG	
highlight	name=btnG	
click	name=btnG	
waitForElementPresent	id=resultStats	
highlight	id=resultStats	
storeText	id=resultStats	Result
echo	**${Result}**	**დაახლოებით 273,000 შედეგი (0.06 წამი)**

./LinkedGTU.htm

LinkedGTU		
setTimeout	60000	
highlight	//html/body/div[2]/div[2]/div/div[4]/div[2]/div[2]/d v[2]/div[2]/div/ol/li/div/h3/a	
click	//html/body/div[2]/div[2]/div/div[4]/div[2]/div[2]/d v[2]/div[2]/div/ol/li/div/h3/a	
pause	10000	
storeLocation	URL	
echo	**${URL}**	**http://www.gtu.edu.ge**

ნახ.3

ავტომატური ტესტირებისას, ზოგადად გვხვდება ორი ტიპის შეცდომები: პირველი, რომელთა აღმოჩენის შემთხვევაში ტესტირების პროცესი ჩერდება და მეორე, რომელთა შემთვევაშიც პროცესი აგრძელებს მომდევნო ბრძანებების შესრულებას [4]. განხილული მაგალითისთვის წარმოვიდგინოთ, რომ გუგლის გახსნისას Selenium-მა ვერ იპოვნა ღილაკი "Google Search", ამ შემთხვევაში ტესტირების პროცეი გაჩერდება, საშედეგო ფაილში ჩაიწერება შეტყობინება, რომ ვერ მოიძებნა ღილაკი "Google Search". ხოლო თუ წარმოვიდგენთ მაგალითს რომ გვინდა აღნიშნული ღილაკის ტექსტის შემოწმება, ემთხვევა თუ არა იგი სტრიქონს "Gooooooogle Search", ამ შემთხვევაშიც მივიღებთ შეცდომას, მაგრამ ტესტირების პროცესი არ გაჩერდება. ამ შემთხვევაში საშედეგო ფაილში ჩაიწერება შეტყობინება: მიმდინარე სტრიქონი - "Google Search" არ ემთხვევა აღწერილ წარწერას - " Gooooooogle Search".

3. დასკვნა

თუ განხილულ უმარტივეს ტესტირების მაგალითს განვიხილავთ თანამედროვე Web აპლიკაციებისთვის, ადვილი მისახვედრია, თუ რაოდენ დიდი შრომის დაზოგვა და საიმედოობის გაზრდა შეუძლია ავტომატურ ტესტირებას. იგი ფართოდ გამოიყენება თანამედროვე ინფორმაციულ ტექნოლოგიებში, კერძოდ ERP, CMS, HR და სხვა დიდი სისტემებისთვის. ბიზნეს მოთხოვნების მიხედვით იწერება სცენარები, რომელთა საფუძველზეც იქმნება ავტომატური სკრიპტები, TestSuite-ბი და მათი მეშვეობით ხორციელდება ავტომატური ტესტირება.

პროგრამის ყოველი ახალი ვერსიის შექმნისას შეგვიძლია გავუშვათ განმეორებადი ავტომატური ტესტები (ე.წ. Regression Testing) და დავადასტუროთ პროგრამის მუშაობის სისწორე, რადგან დავრწმუნდეთ, რომ ვერსიის ცვლილებამ არ დააზიანა პროგრამის სხვა ფუნქციები. შევნიშნოთ, რომ ავტომატური ტესტირება იგივეა, რაც ხელით ტესტირება, მათ შორის განსხვავებაა ის, რომ

ავტომატური ტესტების გაშვება ხორციელდება უფრო სწრაფად, შედეგები იწერება საშედეგო ფაილში, მისი შესრულებაზე გაშვება ხდება ადამიანის ჩარევის გარეშე, მისი გაშვება შეგვიძლია მრავალჯერადად. ყოველივე ეს იწვევს შეცდომების თავიდან აცილებას, ხარისხის ამაღლებას, დროის დაზოგვას და ხარჯების შემცირებას.

ლიტერატურა:

1. Next Generation Java Testing TestNG and Advanced Concepts. 2009
2. Integrating ERP can overcome CRM Limits. Software Magazine. Earls, 2002
3. Swift M. Accelerating Customer Relationships Using CRM and Relationship technologies. 2001
4. Selenium Documentation from Selenium Project. 2003
5. http://en.wikipedia.org/wiki/HP_QuickTest_Professional. ბოლო შემოწმება: 15.04.2012
6. სურგულაძე გ., გულიტაშვილი მ., კაკულია ი., ჩერქეზიშვილი გ., ჯავახიშვილი ი. პროგრამული სისტემების სასიცოცხლო ციკლის პროცესის მოდელირება უნივერსალური და ექსტრემალური პროგრამირების პრინციპების კომპრომისული გადაწყვეტით. სტუ, შრ.კრ. „მას“- №1(8), 2010, გვ.63-70.

AUTOMATED TESTING OF WEB APPLICATION

Gulitashvili Mikheil, Surguladze Gia
Georgian Technical University

Summary

In the represented article issues of program applications' life cycle management, in particular, management of software testing is considered. Classification of testing software systems, including bases of web-applications testing processes automation is offered. The example of development and running of the test-scripts is shown. There are specified spheres of application of automatic testing in modern information technologies.

АВТОМАТИЗИРОВАННОЕ ТЕСТИРОВАНИЕ ВЕБ-АППЛИКАЦИЙ

Гулиташвили М., Сургуладзе Г.
Грузинский Технический Университет

Резюме

Рассматриваются вопросы менеджмента жизненного цикла программных аппликаций, в частности этап тестирования. Предложена классификация систем тестирования программного обеспечения, в том числе и основы автоматизации процессов тестирования веб-апликаций. Демонстрируется пример создания и функционирования тест-скриптов. Выделяются прикладные сферы применения автоматического тестирования в современных информационных технологиях.