საქართველოს ტექნიკური უნივერსიტეტი

ხელნაწერის უფლებით

მიხეილ გულიტაშვილი

ორგანიზაციული მენეჯმენტის IT-კონსალტინგის მოდელების და მეთოდების დამუშავება და კვლევა სისტემების ინტეგრაციის მიზნით

დოქტორის აკადემიური ხარისხის მოსაპოვებლად

წარდგენილი დისერტაციის

ა ვ ტ ო რ ე ფ ე რ ა ტ ი

თბილისი

2014 წელი

სამუშაო შესრულებულია საქართველოს ტექნიკური უნივერსიტეტის ინფორმატიკისა და მართვის სისტემების ფაკულტეტის „მართვის ავტომატიზებული სისტემების (პროგრამული ინჟინერია)“ დეპარტამენტში

სამეცნიერო ხელმძღვანელი: ტექნიკის მეცნიერებათა დოქტორი,
სრული პროფ. გია სურგულაძე

რეცენზენტები:
სრული პროფ. გურამ ცერცვაძე
სრული პროფ. ალეკო ქუთათელაძე

დაცვა შედგება **2014 წლის ” 5 ” ივლისს, 14.00 საათზე**
საქართველოს ტექნიკური უნივერსიტეტის - „ინფორმატიკისა და მართვის სისტემების“ ფაკულტეტის სადისერტაციო საბჭოს კოლეგიის სხდომაზე, კორპუსი 4, აუდიტორია 401
მისამართი: 0175, თბილისი, კოსტავას 77.

დისერტაციის გაცნობა შეიძლება სტუ-ს ბიბლიოთეკაში,
ხოლო ავტორეფერატისა - სტუ-ს ვებგვერდზე

სადისერტაციო საბჭოს
მდივანი: სრული პროფ. თინათინ კაიშაური

ნაშრომის ზოგადი დახასიათება

თემის აქტუალურობა. ორგანიზაციული მენეჯმენტის ბიზნეს-პროცესების მხარდაჭერა ინფორმაციული ტექნოლოგიებით IT-კონსალტინგის ძირითადი ამოცანაა, რომელიც სისტემის ინტეგრაციის მიზნით იხილავს IT-სერვისების დაგეგმვას, დაპროექტებას, დეველოპმენტს, ტესტირებას და დანერგვას, შესაბამისი ხარჯების ოპტიმიზაციით, ბიზნეს-პროცესების ეფექტურობის, მართვადობის და გამჭვირვალობის ამაღლებით, აგრეთვე ერთიანი IT-ინფრასტრუქტურის შექმნით. თანამედროვე საინფორმაციო ტექნოლოგიები გვთავაზობს ბიზნეს–პროცესების მოდელირების, დაპროექტების, პროგრამული რეალიზაციის და ტესტირების მძლავრ და მოქნილ (Agile) მეთოდებს და ინსტრუმენტულ (CASE) საშუალებებს.

ობიექტ-ორიენტირებული მოდელირების მიდგომამ და დაპროგრამების ახალმა პარადიგმებმა საფუძველი ჩაუყარა პროგრამული სისტემების სასიცოცხლო ციკლის პროცესების სრულყოფას. სისტემების შექმნის და/ან გაფართოების მიზნით მნიშვნელოვანი საკითხია ინტეგრაციის პროცესების მართვა. IT-კონსალტინგი განსაკუთრებულ ყურადღებას უთმობს ტესტირების ახალ მეთოდოლოგიებს, ვინაიდან IT-სერვისების (ან ბიზნეს-პროცესების პროგრამული მოდულების) დამუშავების პროცესში ტესტირება და შეფასების განსაზღვრა მეტად მნიშვნელოვანია. აქ წყდება საკითხი დამუშავებული სერვისების საბოლოო გამოყენების ან მათი შემდეგი ვერსიის შექმნის შესახებ, რაც კარგად აისახება პროგრამის სასიცოცხლო ციკლის იტერაციულ-ინკრემენტალურ მოდელზე.

მაგალითად, ავტომატური Regress და Unit ტესტირების მეთოდოლოგია პროგრამული სისტემების შექმნის ავტომატურ პრინციპებზეა აგებული. მაგალითად, ავტომატური Regress და Unit ტესტირება. ორივე ტიპის მეთოდოლოგია პროგრამული სისტემების შექმნის ავტომატურ პრინციპებზეა აგებული. ავტომატური ტესტირება გამოიყენება დიდი და საშუალო პროგრამული პროექტების ვალიდაცია–ვერიფიკაციისთვის, იგი ეფუძნება პროგრამული კოდით მატესტირებელი სცენარების შექმნას და დამხმარე ინსტრუმენტებით არსებული აპლიკაციის დამოწმებას. Agile

მეთოდოლოგიის მიმდევრები (მაგ., ექსტრემალური პროგრამირება (XP), Scrum, Rational Unified Process (RUP), Dynamic Systems Development Method (DSDM) და სხვ.) აქცენტს მაღალორგანიზებულ გუნდური პროგრამირების პრინციპებზე აკეთებენ და ცდილობენ გამორიცხონ პროგრამული ხარვეზების დაშვება განპირობებული ადამიანური ფაქტორებით.

თანამედროვე საინფორმაციო ტექნოლოგიებში ავტომატური ტესტირება ახალი დარგია და იგი დღითი–დღე ხდება პოპულარული. ტესტირება არის პროგრამული უზრუნველყოფის სასიცოცხლო ციკლის შემადგენელი აუცილებელი ეტაპი, რომელიც პირდაპირ განსაზღვრავს აპლიკაციის ხარისხს. ამერიკისა და ევროპის უნივერსიტეტებსა და Software-ის საპროექტო ინსტიტუტებში, დიდი სერიოზული პროექტები მოწმდება ავტომატური მატესტირებელი ტექნოლოგიის პრინციპებით. საქართველოში ეს მიმართულება დამკვიდრების სტადიაშია, თუმცა ისეთი მოწინავე სტრუქტურა როგორიცაა ფინანსთა სამინისტროს ანალიტიკური სამსახური, უკვე აქტიურად იყენებს ტესტირების ავტომატიზაციას შექმნილი პროდუქტების მუშაობის დასამოწმებლად.

ახალი გენერაციის ინფორმაციულ ტექნოლოგიებს მიეკუთვნება კომპანია Microsof-ის მიერ შემუშავებული BDT ტექნოლოგია, რომელიც გულისხმობს აპლიკაციის პროგრამული კოდის ავტომატურ Build-ს, დაბილდული პროექრის ავტომატურ განთავსებას – ე.წ. Deploy და ბოლო ეტაპზე ავტომატურ ტესტირებას, რაც გულისხმობს აპლიკაციაზე ავტომატური ტესტების გადატარებას

ავტომატური ტესტირების ტექნოლოგიების საშუალებით შესაძლებელია ბიზნეს აპლიკაციის სრული სამომხმარებლო ინტერფეისის მუშაობის ემულაცია, შეცდომების პოვნა მანქანური რესურსების დანახარჯებით. აგრეთვე შესაძლებელია აპლიკაციის დატვირთვით მუშაობის შეფასება Performance ტესტირების საშუალებით.

პროგრამულ ინჟინერიაში საინფორმაციო სისტემების დაპროექტების, რეალიზაციასა და შემოწმების თვალსაზრისით, ავტომატური ტესტირება იქცა უნივერსალურ მეთოდოლოგიად, რომელიც განკუთვნილია როგორც

პროგრამული სისტემების ტესტერებისთვის, ისე ბიზნეს-ანალიტიკოსებისთვის და პროგრამისტებისთვის. ამ ტექნოლოგიის აქტიურმა გამოყენებამ აუცილებელი გახადა ავტომატური ტესტირების ინსტრუმენტების განვითარება და თანამედროვე ტექნოლოგიებთან თავსებადი ვერსიების შექმნა.

ნაშრომში წარმოდგენილია ხელით ტესტირებაც, მისი შედარება ავტომატურ ტესტირებასთან, მათი გამოყენების სფეროები, ნაჩვენებია მათი მოხმარების ხელსაყრელი არეები. ყოველივე ზემოთქმული მეტყველებს დისერტაციის თემის აქტუალურობაზე და მის განსაკუთრებულ მნიშვნელობაზე.

სამუშაოს მიზანი და ამოცანები. დისერტაციის მიზანია ორგანიზაციული მენეჯმენტის სისტემების გამოყენებითი პროგრამული აპლიკაციების ავტომატური ტესტირების საკითხების კვლევა, მისი კავშირი სისტემების ინტეგრაციის პრობლემებთან. ობიექტ–ორიენტირებული დაპროგრამებისა და მოქნილი (Agile) პროგრამული რეალიზაციის ახალი ტექნოლოგიებით მანქანური ვალიდაციების, ფასიანი და უფასო მატესტირებელი სისტემების გაცნობა, Selenium ტექნოლოგიის და მისი შემადგენელი ინსტრუმენტები მიმოხილვა, Microsoft-ის მიერ შემოთავაზებული CodedUI და Unit ტესტირების ტექნოლოგიების გამოყენება. აგრეთვე იმ დადებითი მხარეების ჩვენება, რომლებიც ამ ავტომატურ ტესტირებას გააჩნია და მის მრავალ სხვადასხვა სფეროებში წარმატებით გამოყენების შესაძლებლობას განაპირობებს, მათ შორის ქვეყნის სახაზინო ელექტრონული სისტემის და ფინანსთა სამინისტროს სხვა ელექტრონული სისტემების საპრობლემო სფეროებში.

დასმული მიზნის მისაღწევად აუცილებელია შემდეგი ძირითადი ამოცანების გადაწყვეტა:

ორგანიზაციული მენეჯმენტის IT-კონსალტინგის მიზნით საინფორმაციო სისტემების ტესტირების პროცესის დაპროექტების და მოდელირების თანამედროვე მეთოდებისა და ინსტრუმენტული საშუალებების ანალიზი, მათი კლასიფიკაცია ბიზნეს-პროცესების ასახვის შესაძლებლობათა ევოლუციის თვალსაზრისით;

- აუქციონის სისტემის საპრობლემო სფეროს ფუნქციონალურ და არაფუნქციონალურ მოთხოვნილებათა განსაზღვრა, შესაბამისი მძლავრი, მრავალმხრივი და მრავალპლატფორმიანი სისტემის - CodedUI ინსტრუმენტის გამოყენებით .NET პლატფორმაზე;

- ბიზნეს-პროცესების IT-სერვისების სადემონსტრაციო მატესტირებელი ტესტ-სკრიპტების ჩაწერა ავტომატურ რეჟიმში თანამედროვე ტექნოლოგიების ბაზაზე;

- ჩაწერილი ტესტ-სკრიპტების იმპორტ-ექსპორტი სხვადასხვა პროგრამულ ენებზე და შესრულებაზე გაშვებისას წარმოქმნილი პრობლემები, მატესტირებელი სისტემის ნაკლოვანებების გამოააშკარავება ტექნოლოგიური შეუთავსებლობის პრინციპებით;

- მატესტირებელი სერვერის ტექნოლოგიური განხილვა, მისი სტრუქტურის მოდელირება, ტესტირების რეპორტები და ბიზნეს აპლიკაციის ავტომატური ტესტირების ანალიზი;

- პარალელური ტესტირების საჭიროება, მისი ტექნიკური რეალიზება სადემონსტრაციო ამოცანის საშუალებით. მიმდევრობითი ტესტირების შედარება პარალელურ ტესტირებასთან შესრულების დროის ოპტიმიზების მიზნით.

- Regress და Unit ტესტირების ტიპები, მათი გამოყენება თანამედროვე ბიზნეს აპლიკაციებში. ავტომატური ტესტირების კლასიფიკაცია.

- ტესტ-ქეისების, ტესტ-სკრიპტების, სცენარების და ტესტირების ძირითადი ტერმინების განმარტება, მათი რეალიზება სხვადასხვა დაპროგრამების ენებით და სკრიპტებით.

- ობიექტზე-ორიენტირებული მეთოდოლოგიის გამოყენება ავტომატურ ტესტირებაში Java ტექნოლოგიის საფუძველზე. მატესტირებელ სცენარებში JavaScript სკრიპტინგ ენის გამოყენება Web აპლიკაციების ტესტირებისას.

- SilverLight ტექნოლოგიის მატესტირებელი plugin-ი .NET პლატფორმაზე, ბიზნეს აპლიკაციის ტესტირების შედეგების ანალიზი ვიდეო ჩანაწერით, Screenshot-ით და პროგრამული კოდით.

კვლევის ობიექტები. ფინანსთა სამინისტროს ელექტრონული სახელმწიფო ვალის მართვისა და აუქციონის სისტემების ტესტირების ავტომატიზებული BDT სისტემა. ინფორმაციული და პროგრამული უზრუნველყოფის ინფრასტრუქტურა და პროგრამული რეალიზაცია.

კვლევის მეთოდები. ავტომატური ტესტირების მეთოდი. ობიექტ-ორიენტირებული დაპროგრამების, სკრიპტინგ ენების ტესტირების მეთოდები. მიმდევრობითი და პარალელური ტესტირების პროცესები და მათი რეალიზაციის ინსტრუმენტები. საინფორმაციო სისტემების ტესტირების კლასიფიკაცია და მათი გამოყენების თეორიული არეები. ბიზნეს-აპლიკაციების ტესტირების თანამედროვე ახალი ტექნოლოგიები. სცენარების ჩაწერის პროგრამული ინსტრუმენტები.

მეცნიერული სიახლე. ორგანიზაციული მენეჯმენტის IT-კონსალტინგის მოდელების და მეთოდების შესამუშავებლად სისტემების ინტეგრაციის მიზნით გამოკვლეულ იქნა ბიზნეს-პროცესების (IT-სერვისების) პროგრამული სისტემების სასიცოცხლო ციკლის მართვის საკითხები, კერძოდ განტერის იტერაციულ-ინკრემენტალური „ფაზა-ფუნქციების“ მოდელის საფუძველზე შესწავლილ იქნა IT-სერვისების ტესტირების შედეგების დამოკიდებულება ინტეგრაციის პროცესებზე. მაგალითად, წარმოდგენილ იქნა ფინანსთა სამინისტროს მომსახურების სააგენტოს ელექტრონული eAuction სისტემის ტესტირების ავტომატიზაციის შედეგები თანამედროვე ტექნოლოგიის გამოყენებით, მისი შემდგომი პროგრამირების და პროცესების ავტომატიზაციის მიზნით, კერძოდ:

1. პირველად დისერტაციაში შემოთავაზებულ იქნა ელექტრონული აუქციონის ერთიანი სისტემის ტესტირების ავტომატიზაციის თანამედროვე კონცეფციები და მის საფუძველზე ვალიდაცია-ვერიფიკაციის საკითხები;

2. განხორციელდა მსგავსი საპრობლემო სფეროების ტესტირების პრობლემების და ამოცანების ჩამოყალიბება, საინფორმაციო სისტემების კლასიფიკაცია და დაპროექტების და მოდელირების შესაბამისი ტექნოლოგიების განსაზღვრა მათ გადასაწყვეტად;

3. პირველად ნაშრომში წარმოდგენილ იქნა ელექტრონული სისტემების ავტომატური ტესტირების პლატფორმა და ავტომატური ტესტების პროგრამული რეალიზაციის ამოცანის გადაწყვეტა ახალი Microsoft, Selenium, პროცეს-ორიენტირებული და ობიექტ-ორიენტირებული მეთოდებით და ინსტრუმენტული საშუალებებით;

შედეგების გამოყენების სფერო. დისერტაციის შედეგებს აქვს პრაქტიკული ღირებულება, ვინაიდან ის შეიძლება გამოყენებულ იქნას სხვადასხვა დარგებში ბიზნეს აპლიკაციების ტესტირების, ვერიფიკაციის, ვალიდაციის და პროგრამული რეალიზაციის ამოცანების გადასაწყვეტად ახალი ინფორმაციული ტექნოლოგიებით.

ნაშრომის აპრობაცია: დისერტაციის ძირითადი შინაარსი მოხსენებული იყო ინფორმატიკისა და მართვის სისტემების ფაკულტეტის „მართვის ავტომატიზებული სისტემების (პროგრამული ინჟინერია)“ კოლეგიის სამეცნიერო სემინარების სხდომებზე, ასევე საერთაშორისო კონფერენციებზე „მართვის ავტომატიზებული სისტემები და ახალი ინფორმაციული ტექნოლოგიები“, MIS-2011, 19-21 მაისი, სტუ, თბილისი; „ინტერნეტი და საზოგადოება“ INSO-2013, 6–7 ივნისი, აკაკი წერეთლის სახელმწოფი უნივერსიტეტი, ქუთაისი. პუბლიკაციები: დისერტაციის ძირითადი შედეგები გამოქვეყნებულია 8 სამეცნიერო ნაშრომში, რომელთა ჩამონათვალიც მოყვანილია დისერტაციის ბოლოს.

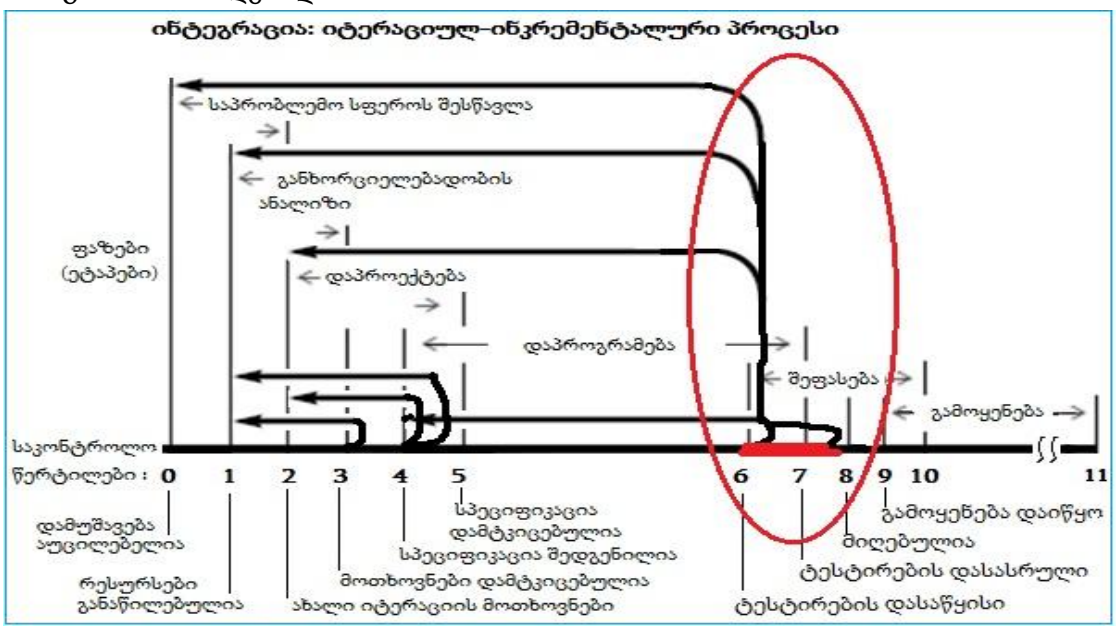
ნაშრომის მოცულობა და სტრუქტურა: დისერტაციის სრული მოცულობა შეადგენს 140 ნაბეჭდ გვერდს; შედგება რეზიუმეს (ორ ენაზე), სარჩევის, შესავლის, სამი თავის და დასკვნისგან. ახლავს 78 ნახაზი და 41 გამოყენებული ლიტერატურის სია.

დისერტაციის მოკლე შინაარსი

შესავალში გადმოცემულია დისერტაციის ზოგადი დახასიათება, თემის აქტუალურობა, მიზანი და გადასაწყვეტი ამოცანები, სამეცნიერო სიახლე და პრაქტიკული ღირებულება. აგრეთვე ნაშრომის მოკლე შინაარსი თავების მიხედვით.

პირველი თავი. სახელმწუფო ორგანიზაციები და ბიზნესის კერძო სტრუქტურები თითქმის მთლიანად კომპიუტერიზებულია. დღევანდელ IT-ბაზარზე მნიშვნელოვანია საუკეთესო პროგრამული პროდუქტის მოპოვება, მაგრამ ეს არც ისე მარტივია. არსებობს მრავალი სხვადასხვა ხელმისაწვდომი პროგრამა, რომელთაგანაც საუკეთესო აპლიკაციებს აფასებენ ხარისხით. ორგანიზაციაში IT-კონსალტინგმა უნდა განსაზღვროს ეს საკითხი ბიზნეს-პროცესების ეფექტური მხარდაჭერის თვალსაზრისით.

პროგრამული უზრუნველყოფის შექმნის პროცესის მართვა მისი სასიცოცხლო ციკლის საფუძველზე, მათ შორის ტესტირების ეტაპიც (საკონტროლო წერტილი იხ.ნახ.1) უნდა გამოვიყენოთ ყოველთვის, საიმედო პროგრამის მისაღებად.



ნახ.1-ბ. პროგრამული სისტემის სასიცოცხლო ციკლი ტესტირებით

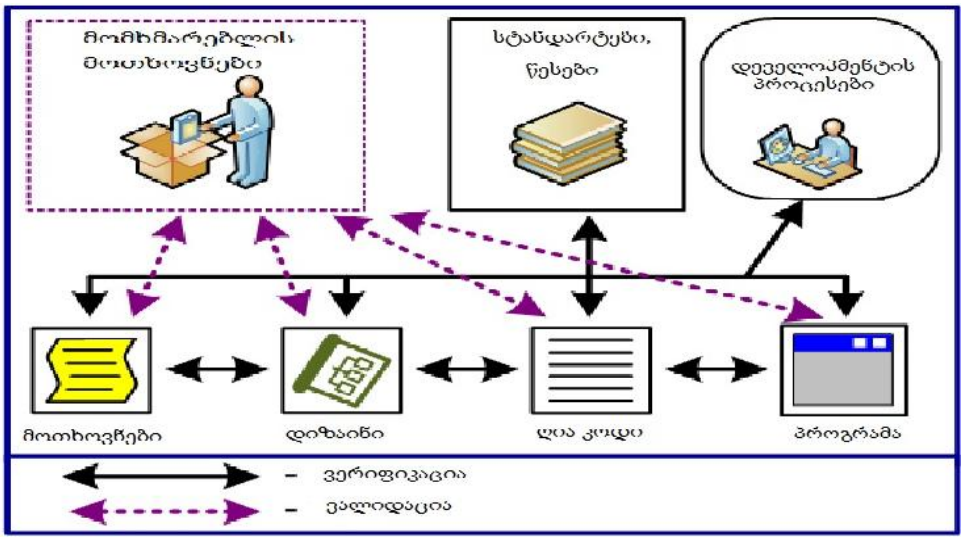
პროგრამული პაკეტების ტესტირება არის ის პროცესი, რომელიც იკვლევს აპლიკაციის ფუნქციებს, რათა მივიღოთ ინფორმაცია პროდუქტის ხარისხზე. პროგრამის ტესტირება გვიჩვენებს სოფტის ობიექტურ სახეს. ტესტირება მოიცავს პროგრამული აპლიკაციის შესრულებაზე გაშვების პროცესს, რომელიც გამიზნულია იპოვნოს პროგრამული შეცდომა ან დეფექტი. ეს თავი ეხება პროგრამული უზრუნველყოფის ტესტირების თანამედროვე ტექნოლოგიების მიმოხილვას. გადმოცემულია ტესტირების, ვალიდაციის და ვერიფიკაციის პროცესების ძირითადი ცნებები, Web აპლიკაციების ავტომატური ტესტირების არსი. ავტომატური ტესტირების წარმოშობის ისტორია და განვითარება. აღწერილია ხელით ტესტირების შედარება ავტომატურ ტესტირებასთან, მათი უარყოფითი და დადებითი მხარეები. განხილულია ტესტირების ძირითადი პრინციპები და კონცეფციები. შემოთავაზებულია პროგრამული ტესტირების კლასიფიკაცია და მოკლედ მიმოხილულია ძირითადი ტესტირების ტიპები.

განხილულია ბაგის წარმოშობის ისტორია და აღწერილია მისი არსებობის გამო წარმოქმნილი წინააღმდეგობები.

ვალიდაცია არის პროცესი, რომლის დროსაც მოწმდება რომ პროდუქტის დიზაინი აკმაყოფილებს, ეთანადება დასმულ მოთხოვნებს, ანუ პროგრამული უზრუნველყოფა ეთანადება მომხმარებლის მოთხოვნებს.

ვერიფიკაცია არის პროცესი, რომლის დროსაც პროგრამული უზრუნველყოფა მოწმდება შექმნის პროცესში, ანუ აკმაყოფილებს თუ არა წინასწარ განსაზღვრულ სტანდარტულ მოთხოვნებს.

დღესდღეობით არსებული პროგრამული უზრუნველყოფის უდიდესი ნაწილი არის Web-ზე ბაზირებული პროგრამული პროდუქტი, რომელთაც მოვიხმართ ინტერნეტ ბრაუზერების საშუალებით (ნახ.2).



ნახ.2 ვერიფიკაცია და ვალიდაცია

ეპოქაში, სადაც არის მაღალი საპასუხისმგებლო პროგრამული უზრუნველყოფის პროცესები, სადაც მრავალი ორგანიზაციები იყენებენ Agile მეთოდოლოგიას, ტესტირების ავტომატიზაცია ხდება დღითიდღე პოპულარული. პროგრამული პროდუქტის ეფექტურ ტესტირებას დიდ მნიშვნელობას ანიჭებენ კომპანიები და ორგანიზაციები. პროგრამული უზრუნველყოფის ტესტირების ავტომატიზაცია ნიშნავს გამოიყენო სხვა პროგრამული ინსტრუმენტი რომლის საშუალებითაც მოახდენ სატესტირებელი პროგრამის მრავალჯერად ტესტირებას. ტესტირების ავტომატიზაციას გააჩნია სხვადასხვა უპირატესობები, რომელთაგანაც აღსანიშნავია მისი **განმეორებადობა და სიჩქარე**, რომელიც საჭიროა ტესტის გასაშვებად.

Selenium არის უფასო პროგრამული უზრუნველყოფა, განკუთვნილი Web აპლიკაციების ავტომატური ტესტირებისთვის. Selenium ტესტები შესაძლებელია შეიქმნას ისეთ თანამედროვე პროგრამირების ენებზე

როგორიცაა: C#, Java, Groovy, Perl, PHP, Python და Ruby. იგი თავსებადია Windows, Linux, Macintosh ოპერაციულ სისტემებთან.

- **Black box testing** (შავი ყუთის ტესტირება) – ტესტირების მეთოდი, რომლის დროსაც ხდება აპლიკაციის ფუნქციონალის ტესტირება შიგა სტრუქტურის გამოკვლევის გარეშე. ამ დროს პროგრამული კოდის ან არქიტექტურის ცოდნა არ არის საჭირო.

"შავი ყუთის" ტიპის ტესტირებისას ტესტერი ამოწმებს თუ რა გააკეთა პროგრამამ და არ აინტერესებს თუ როგორ მიიღო შედეგი.

მაგალითად, კომპიუტერულ პროგრამას შესასვლელზე ვაძლევთ რაღაც მონაცემებს, ანუ ვაწვდით Input-ს, ხდება ამ მონაცემების დამუშავება და შედეგად ვიღებთ Output-ს (ნახ.3). აპლიკაციის შიგა პროცესები უგულებელყოფილია.

საბოლოოდ, შეგვიძლია დავასკვნათ, რომ "შავი ყუთის" ტესტირება არის ტესტირების ტიპი, რომლის დროსაც ხდება პროგრამის ფუნქციონალის შემოწმება შიგა სტრუქტურის გამოკვლევის გარეშე.

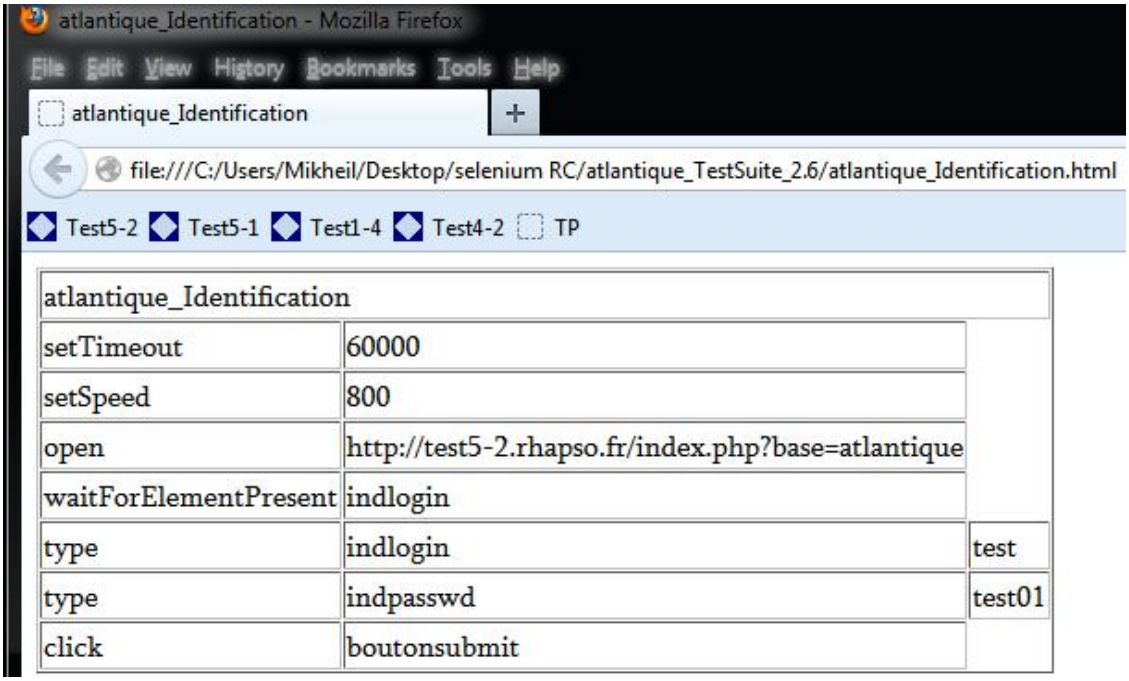
ნახ.3 Black box testing

- **White box testing** (თეთრი ყუთის ტესტირება) – კომპიუტერული პროგრამის ტესტირების მეთოდი, რომლის დროსაც ტესტირების პროცესი მიმდინარეობს პროგრამული უზრუნველყოფის შიგა სტრუქტურის დონეზე.

"თეთრი ყუთის" ტესტირება მოითხოვს პროგრამის შიგა სტრუქტურის, პროგრამული კოდის ცოდნას. ძირითადად, "თეთრი ყუთის" ტესტირებას ახორციელებენ პროგრამისტები, რადგან როგორც აღვნიშნეთ მის დროს აპლიკაციის შემოწმება ხდება პროგრამული კოდის დონეზე. მსგავსი ტიპის ტესტირება დაფუძნებულია პროგრმულ კოდის ცოდნაზე, კომპონენტებზე და მათ შორის კავშირებზე, ციკლებზე, და სხვ.

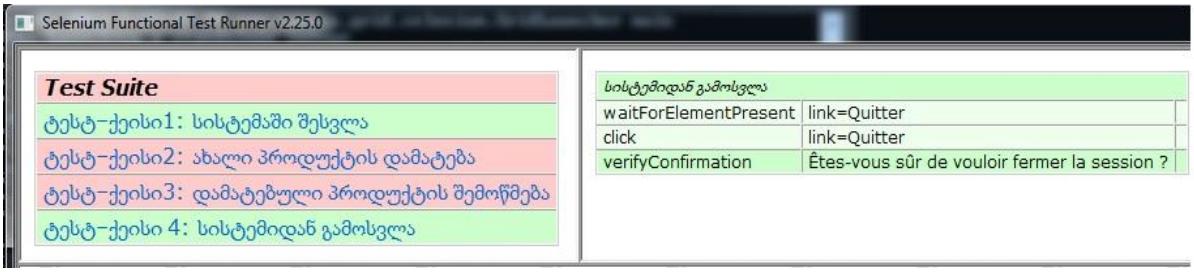
პროგრამული უზრუნველყოფის ტესტირებისას შესაძლებელია დაიწეროს ხელოვნური ტესტ–სკრიპტი, რომელიც შესრულდება ადამიანის მიერ, ან შესაძლებელია გაკეთდეს ტესტ–ქეისში მოცემული ინსტრუქციების ავტომატიზაცია. ქვემოდ წარმოდგენილ ნახ.4–ზე განხილულია Selenium ტესტი, რომელიც წარმოადგენს ავტომატურ ტესტ–სკრიპტებს (იხ. ნახ.5).

ტესტ–სუიტი არის ტესტ–ქეისების ერთობლიობა. იმისთვის რომ დატესტირდეს არსებული სისტემის ერთი რაღაც მთლიანი ფუნქცია, ამისთვის ტესტ–ქეისები ერთიანდება სხვადასხვა ჯგუფებად, ტესტ–სუიტებად.



ნახ.4. ტესტ–სკრიპტი

სცენარებში შემავალი სკრიპტები შეიძლება იყოს ერთმანეთის შესრულებაზე დამოკიდებულნი. ამ შემთხვევაში ამბობენ რომ სუიტში შემავალი ტესტ–ქეისები დამოკიდებულნი არიან ერთმანეთზე, ანუ შემდგომი ტესტ–სკრიპტის შესრულება დამოკიდებულია წინა ტესტ–სკრიპტის შესრულების შედეგზე (იხ. ნახ.5).



ნახ.5. ტესტ–სუიტი

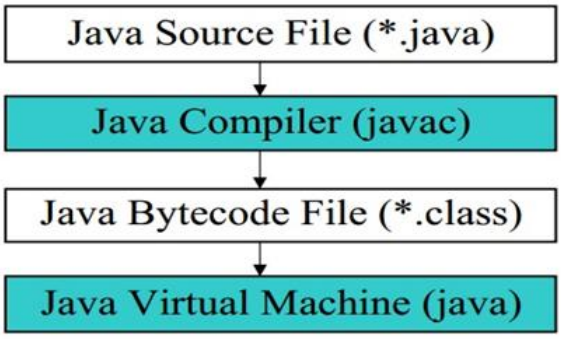
მწვანე ფერით მოცემულია წარმატებით შესრულებული ტესტ–სკრიპტები, ხოლო წითელ ფერში წარმოდგენილია ჩავარდნილი ტესტები. მოცემული სურათიდან შეგვიძლია დავასკვნათ, რომ ჩავარდა „ახალი პროდუქტის დამატების“ ტესტი, შესაბამისად ჩავარდა მასზე დამოკიდებული მომდევნო ტესტი – „დამატებული პროდუქტის შემოწმება“, ხოლო წარმატებით შესრულებულა სისტემაში „შესვლა – გამოსვლის“ ტესტები.

მეორე თავში ფართოდაა მიმოხილული Web-აპლიკაციების ავტომატური მატესტირებელი სისტემა Selenium, მისი შემადგენელი კომპონონტები და მატესტირებელი ინსტრუმენტები. განხილულია

მატესტირებელი სცენარის ჩაწერა, ტესტ–სკრიპტების და ტესტ–სცენარების შექმნა მოცემული ტექნოლოგიით. აღწერილია ტესტირების ძირითადი ქმედებების, ვალიდაციის და ვერიფიკაციის ბრძანებები. გამოკვლეულია AJAX ტექნოლოგიით შექმნილი აპლიკაციების ტესტირებისას წარმოქმნილი პრობლემები ავტომატური ტესტის შესრულების თვალსაზრისით. მიმოხილულია აპლიკაციის ელემენტებზე წვდომის გზები – ლოკატორები, მათი კლასიფიკაცია და ერთმანეთთან შედარება ტესტის სწრაფად შესრულება–სტაბილურობის თვალსაზრისით. განხილულია ავტომატურ ტესტებში სკრიპტინგ ენებისა და ობიექტზე ორიენტირებული ენების გამოყენება და მათი უპირატესობები.

თვალსაჩინოებისთვის, აღნიშნული მატესტირებელი სისტემებისათვის დამუშავებულია სადემონსტრაცი მაგალითები რომელთაც დიდი პრაქტიკული გამოყენება აქვთ. ობიექტზე ორიენტირებული დაპროგრამების ენებით რეალიზებულია მატესტირებელ ბრძანებათა სისტემის გაფართოება ავტომატური ტესტირების გამარტივების თვალსაზრისით.

Java პლატფორმაზე პროგრამირებისას Java–ს პროგრამული კოდი იწერება .java ფაილში. მის შესრულებაზე გაშვებისას კომპილატორი(javac) ამოწმებს კოდის სინტაქსურ სისწორეს, დაკომპილირებულ კოდს გარდაქმნის **ბაიტ–კოდებად** და წერს .class ფაილებში. ბაიტ–კოდების შესრულებაზე გაშვება ხდება **JVM**-ის(Java virtual machine) საშუალებით, რომელიც .class ფაილებში ჩაწერილ ბაიტ–კოდებს გადაიყვანს პროცესორისთვის გასაგებ მანქანურ კოდებში(ნახ.6). ეს განასხვავებს Java კომპილატორს სხვა პროგრამირების ენის კომპილატორებისგან, რომლებიც პროგრამულ კოდს პირდაპირ გარდაქმნიან მანქანურ კოდებში.



ნახ.6. Java: .java ფაილის კომპილაცია

Java პლატფორმის ძირითადი შემადგენელი ნაწილებია ***Java Runtime Environment(JRE)*** და ***Java Development Kit(JDK)***. JDK შეიცავს პროგრამული დეველოპმენტისთვის საჭირო ინსტრუმენტებს: Java–ს კომპილატორი - **javac**(რომელიც Java კოდს გარდაქმნის ბაიტ–კოდებში), **javadoc** - დოკუმენტაციის გენერატორი, **JAR** – არქივის ინსტრუმენტი, რომელიც

კლასთა ბიბლიოთეკებს აარქივებს JAR ფაილად(აღნიშნული ინსტრუმენტი ასევე მართავს JAR ფაილებს), სრულ კლასთა ბიბლიოთეკას რომელთა გამოყენებით შესაძლებელია ნებისმიერი აპლიკაციის შექმნა.

Java Runtime Environment(JRE) შეიცავს JVM–ს(ჯავას ვირტუალური მანქანა), Java კლასთა ბიბლიოთეკებს, Java ენაზე დაწერილი პროგრამების გასაშვებად საჭირო კომპონენტებს. აღსანიშნავია რომ JDK მთლიანად შეიცავს JRE-ს. ანუ JRE–ში შემავალი კმპონენტები შედიან JDK-ში.

ავტომატური ტესტების წარმატებით მუშაობა დამოკიდებულია GUI (Graphical Use Interface) ელემენტების იდენტიფიკაციაზე, მათი განლაგების განსაზღვრაზე, რათა ტესტის შესრულებისას მათზე განხორციელდეს სხვადასხვა ოპერაციები. Selenium–ს გააჩნია სხვადასხვა ინსტრუმენტი რომლებიც განსაზღვრავს ელემენტებზე წვდომის განსხვავებულ გზებს: Name, ID, Link, CSS და Xpath.

Web დეველოპმენტის პროექტში რეკომენდებულია GUI ელემენტებისთვის აღიწეროს ისეთი ატრიბუტები, როგორიცაა: ID, Name, Class. აღნიშნული ატრიბუტები იძლევა იმის შესაძლებლობას, რომ აპლიკაცია გახდეს უფრო ტესტირებადი და სტანდარტული გზებით მოხდეს ელემენტებზე წვდომა.

მიუხედავად ამისა პრაქტიკაში გვხვდება ისეთი შემთხვევები, სადაც შეუძლებელია მოცემული ატრიბუტების განსაზღვრა, ან ელემენტს უბრალოდ არ აქვს აღწერილი ID, Name. მსგავს შემთხვევებში იყენებენ CSS და XPath ლოკატორებს.

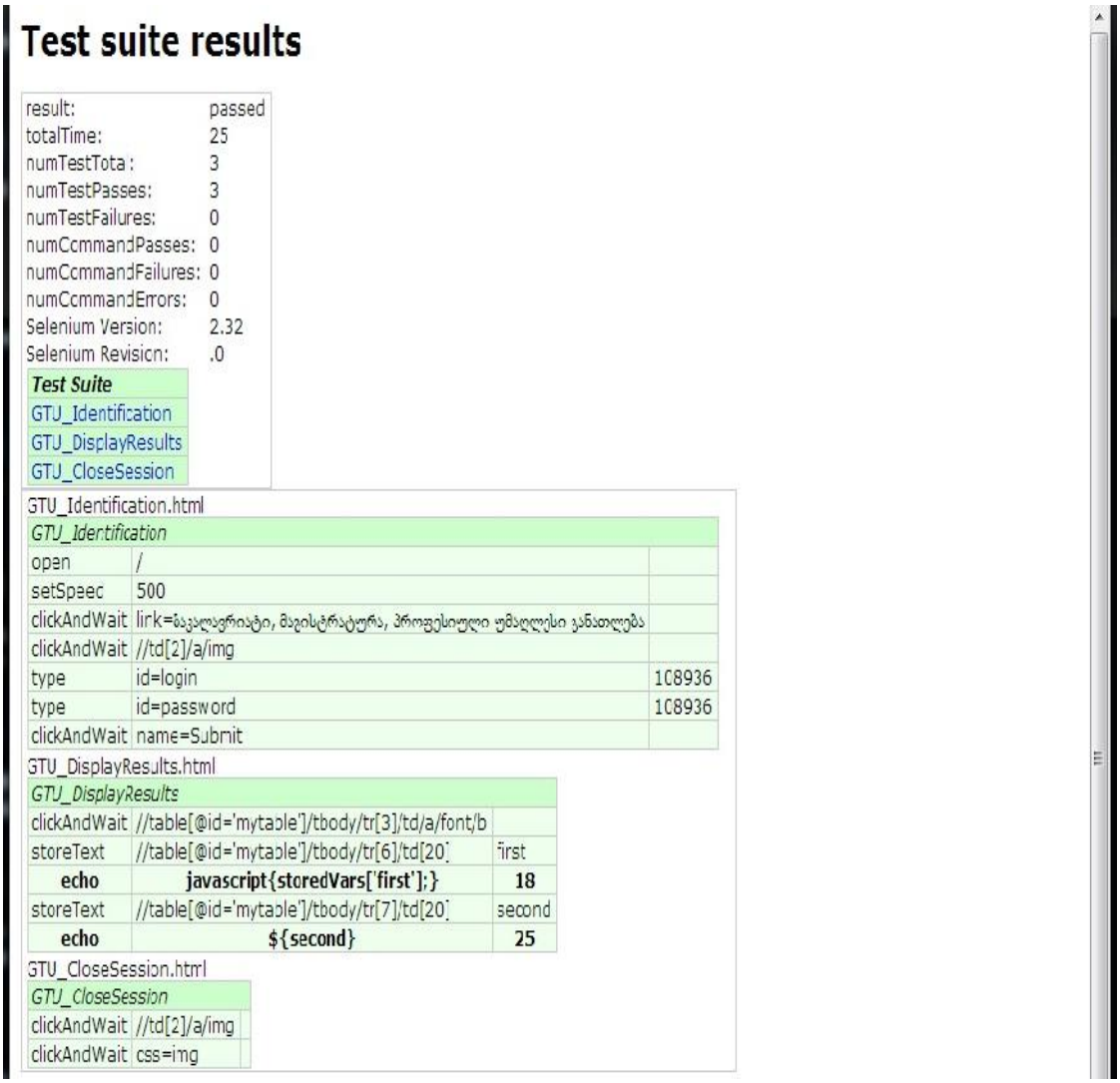
Web გვერდზე არსებული ელემენტის ლოკატორი არის მისი მისამართი, რომლითაც მივმართავთ ელემენტს: ID, name, link, XPath, CSS. ლოკატორები გამოიყენება ტესტის წერისას, რათა მოვახდინოთ საჭირო ელემენტის იდენტიფიკაცია და განლაგების განსაზღვრა. Selenium ტესტში ლოკატორის საშუალებით მივმართავთ ელემენტს მასზე გარკვეული ურთიერთქმედების მიზნით.

Selenium სისტემა დაწერილია JavaScript ტექნოლოგიით. აღნიშნულის გამო შესაძლებელია web გვერდზე მანიპულირება JS სტანდარტული ფუნქციებით. ეს გულისხმობს იმას, რომ თუ Selenium ინსტრუმენტი არ გვთავაზობს საჭირო ფუნქციას, შესაძლებელია ტესტებში წმინდა JavaScript-ის გამოყენება.

Selenium ტესტ–სკრიპტებში JS ფუნქციების გამოყენებას მიმართავენ მაშინ, როდესაც დასატესტირებელია რთულად ტესტირებადი სცენარები. თუ ტექნოლოგია არ არის თავსებადი JavaScript–თან, მაშინ Selenium–ი უძლურია მოცემული სცენარის რეალიზაციისთვის, ამ შემთხვევაში

მიმართავენ უფრო მძლავრი ინსტრუმენტების გამოყენებას, როგორებიცაა: **CodedUI** და **Test Complete.**

მოცემულ ნახაზზე წარმოდგენილია Selenium RC ის მიერ დაგენერირებული საშედეგო ფაილი(ნახ.7).



Test suite results

result:	passed
totalTime:	25
numTestTota :	3
numTestPasses:	3
numTestFailures:	0
numCcmmandPasses:	0
numCcmmandFailures:	0
numCcmmandErrors:	0
Selenium Version:	2.32
Selenium Revisicn:	.0

Test Suite
GTU_Identification
GTU_DisplayResults
GTU_CloseSession

GTU_Identification.html

GTU_Identification		
open	/	
setSpeec	500	
clickAndWait	lirk=ბაკალავრიატი, მაგისტრატურა, პროფესიული უმაღლესი განათლება	
clickAndWait	//td[2]/a/img	
type	id=login	108936
type	id=password	108936
clickAndWait	name=Submit	

GTU_DisplayResults.html

GTU_DisplayResults		
clickAndWait	//table[@id='mytable']/tbody/tr[3]/td/a/font/b	
storeText	//table[@id='mytable']/tbody/tr[6]/td[20]	first
echo	**javascript{storedVars['first'];}**	**18**
storeText	//table[@id='mytable']/tbody/tr[7]/td[20]	second
echo	**${second}**	**25**

GTU_CloseSession.html

GTU_CloseSession	
clickAndWait	//td[2]/a/img
clickAndWait	css=img

ნახ.7. Selenium RC: ტესტის რეპორტ ფაილი

შევნიშნოთ რომ თუ რომელიმე სკრიპტი ჩავარდება, ანუ ვერ გაივლის ტესტს, იგი საშედეგო ფაილში მოიცემა წითელი ფერით.

Selenium WebDriver არის Web აპლიკაციების ავტომატური ტესტირების ფრეიმვორკი, რომელსაც შეუძლია გაუშვას ტესტები **სხვადასხვა ბრაუზერზე** (ნახ.8) და **სხვადასხვა მოწყობილობაზე**.

ნახ.8. WebDriver–ის ბრაუზერებთან თავსებადობა

WebDriver აგრეთვე საშუალებას გვაძლევს შევქმნათ ტესტები თანამედროვე პროგრამირების ენის გამოყენებით, მაშასადამე:

- შესაძლებელია პირობითი ოპერატორების გამოყენება: if-then-else ან switch-case
- შესაძლებელია ციკლების გამოყენება: for, while, do-while

WebDriver-ის მიერ მხარდაჭერილი პროგრამირების ენებია: Java, .Net, PHP, Python, Perl და Ruby. ჩამოთვლილთაგან არ არის საჭირო ყველა ენის ცოდნა. საკმარისია ჩამოთვლილთაგან ვიცოდეთ ერთი რომელიმე ენა. ჩვენ განვიხილავთ Java პროგრამირების ენას და Eclipse პროგრამირების გარემოს.

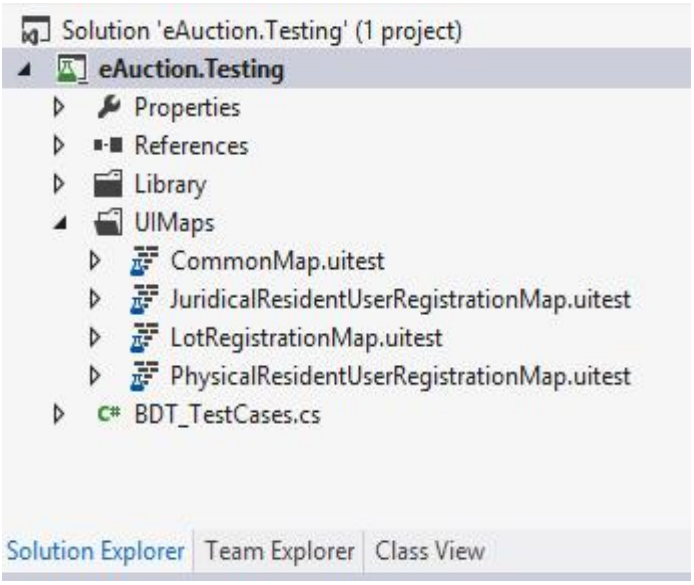
Coded UI ტესტი წარმოადგენს Microsoft ტესტირების ინსტრუმენტს, რომელიც ავტომატურად იწერს, შესრულებაზე უშვებს და ამოწმებს ტესტ-ქეისებს. განსხვავებით სხვა ავტომატური ტესტებისგან(როგორიცაა Unit ტესტი) Coded UI ტესტი მუშაობს სამომხმარებლო ინტერფეისის დონეზე და ტესტავს აპლიკაციას როგორც ადამიანი მაუსით და კლავიატურით ხელში. Coded UI ტესტის ობიექტზე-ორიენტირებული პროგრამირების ენით შესაძლებელია UI ელემენტების არსებობის და მათი მნიშვნელობების დამოწმება.

Coded UI ტესტების წერა შესაძლებელია C# ან Visual Basic-ზე Visual Studio გარემოში. მას გააჩნია ტესტ-ქეისების ჩამწერი ინსტრუმენტი - Coded UI Test Builder, რომელიც აგენერირებს C# ან Visual Basic პროგრამულ კოდს. Coded UI ტესტირების ტექნოლოგია განვიხილოთ ფინანსთა სამინისტროს ელექტრონული აუქციონის eAuction სისტემის ტესტირების მაგალითზე.

განვიხილოთ eAuction ელექტრონული აუქციონი და მისი ავტომატური ტესტირების პროექტის BDT(Build, Deploy, Test) გარემო.

ვთქვად საჭიროა დაიწეროს მომხმარებლის რეგისტრაციის ტესტი eAuction საიტზე. ტესტი უნდა იყოს ავტომატური და მრავალჯერადი გამოყენების(Regress Test). ავტომატური ტესტის მრავალჯერადი გაშვებისთვის აუცილებელია სცენარით დარეგისტრირებული მომხმარებელი ტესტის დასრულების ბოლოს წაიშალოს ბაზიდან, რადგან ტესტის მეორედ გაშვებისას აღწერილი მომხმარებელი უკვე იარსებებს სისტემაში და აპლიკაცია გამოგვიტანს შეტყობინებას: „მითითებული მოხმარებელი უკვე რეგისტრირებულია“.

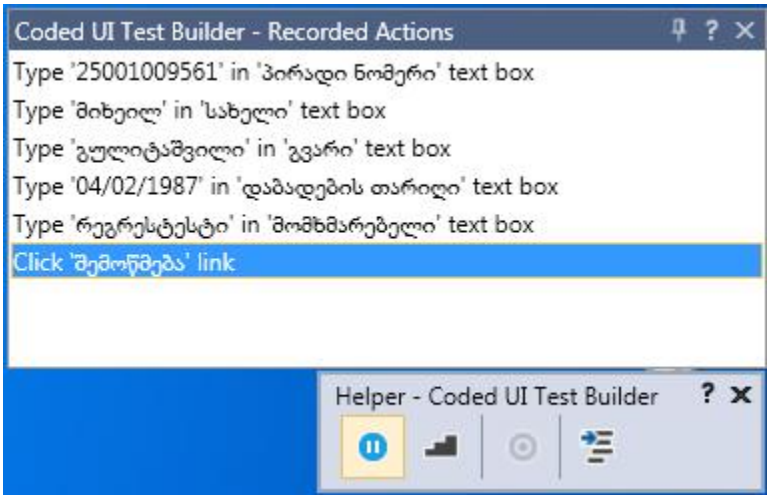
მოცემული სცენარის რეალიზაციისთვის საჭიროა Visual Studio-ში შევქმნათ CodedUI ტესტ-პროექტი ზემოთ აღწერილი ინსტრუქციის საფუძველზე.



ნახ.9. Solution Explorer: Coded UI ტესტ-პროექტი Visual Studio გარემოში

შექმნილ პროექტში საჭიროა დავამატოთ .uitest გაფართოების ფაილი, რომელიც წარმოადგენს სამომხმარებლო ინტერფეისის “რუკას“. Uitest ფაილის იმპლემენტაცია ხდება Test Builder ინსტრუმენტის საშუალებით.

„მომხმარებლის რეგისტრაცია“ სცენარის რეალიზებისთვის საჭიროა, რომ Test Builder ინსტრუმენტი გავუშვათ ჩაწერის რეჟიმში და ჩავიწეროთ ჩვენი ყოველი ურთიერთქმედება სამომხმარებლო ინტერფეისთან, როგორიცაა: მომხმარებლის დასახელების შეყვანა, პირადი ნომერის ჩაწერა, შემოწმება ღილაკზე დაჭერა და ა.შ. (იხ. ნახ.10).

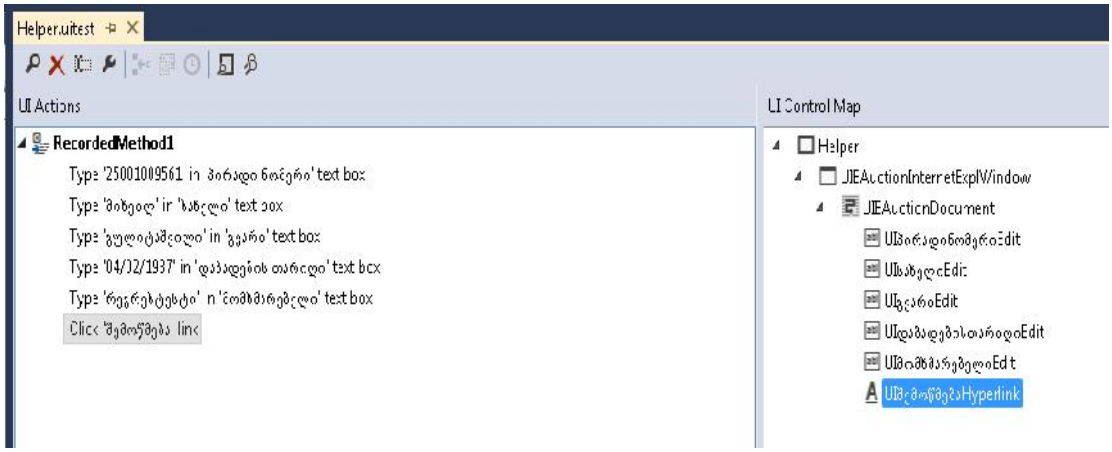


ნახ.10. Test Builder: ჩაწერილი სცენარის ფრაგმენტი

ჩაწერილი სცენარის ფრაგმენტი .uitest ფაილში წარმოდგება შემდეგი ხისებური სტრუქტურით:

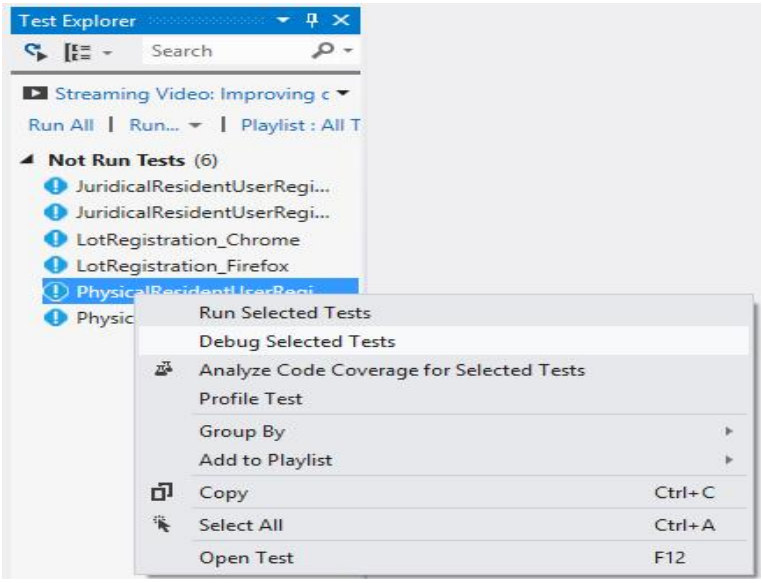
მოცემულ ნახაზს თუ დავაკვირდებით, ყოველ Web ელემენტზე წვდომა ხორციელდება ხისებური სტრუქტურით: ჯერ ვპოულობთ მშობელ ელემენტს, მერე გადავდივართ მის შვილ ელემენტზე და ეს პროცესი გრძელდება მანამ, სანამ არ ჩავალთ სცენარით განსაზღვრულ ელემენტზე.

Uitest ფაილი ავტომატურად აგენერირებს პროგრამულ კოდს. მასში ჩაწერილი სცენარის ცვლილება შესაძლებელია ობიექტ-ორიენტირებული დაპროგრამების გამოყენებით, მაგ: C#, Visual Basic(იხ ნახ.11). თუ დავაკვირდებით მოცემულ სუათს, Test Suilder-ი სცენარის ყოველ ბიჯს ავტომატურად უკეთებს კომენტარს.



ნახ.11. uitest ფაილი: ინტერფეისის რუკა

Visual Studio საშულებით შესაძლებელია ჩაწერილი „მომხმარებლის რეგისტრაციის“ სცენარის შესრულებაზე გაშვება-Run ან შემოწმების პროცესის დაწყება-Debug, Test Explorer ინსტრუმენტით, იხ. ნახ.12:



ნახ.12. Test Explorer ფანჯარა

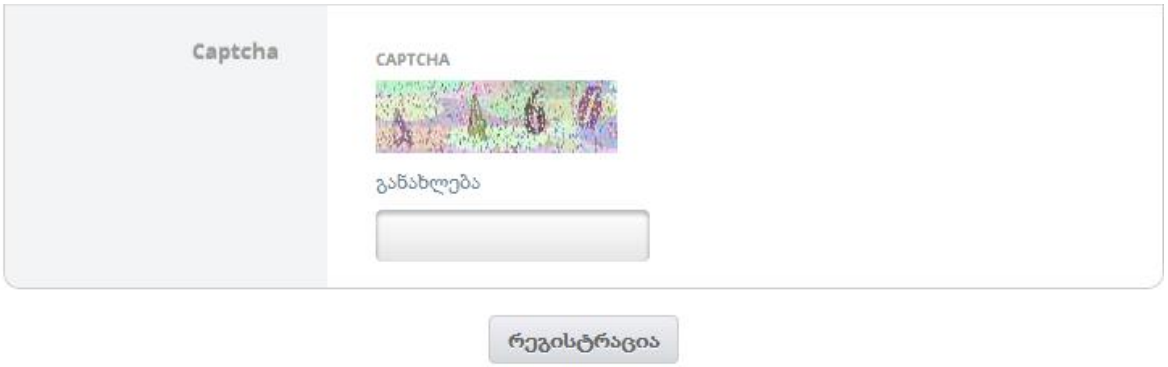
შესრულებაზე გაშვებული ტესტი ჩავარდნის შემთხვევაში გაწითლდება, ხოლო წარმატებით მუშაობისას იგი გამწვანდება, საილუსტრაციოდ იხილეთ Test Explorer ფანჯარა მოცემულ ნახ.13-ზე.

შევნიშნოთ რომ შესრულებაზე არგაშვებული ტესტი Test Explorer-ზე მოიცემა ლურჯი ფონით.

ნახ.13. Test Explorer: წარმატებით დასრულებული და ჩავარდნილი ტესტი

შენიშვნა: უმეტეს საიტზე რეგისტრაციისას საჭიროა Captcha - ზე არსებული სიმბოლოების ამოცნობა და შესაბამის ველში შეყვანა. Captcha სიბოლოების ამოცნობა შეუძლია მხოლოდ ადამიანს, აღნიშნულის გამო ავტომატური ტესტი **Captcha შემოწმებას ვერ გაივლის**.

ზემოაღნიშნულის გამო, eAuction სისტემის „მომხმარებლის რეგისტრაციის“ ავტომატური ტესტი არ იმუშავებს, რადგან სიმბოლოების შეყვანისას ჩავარდება



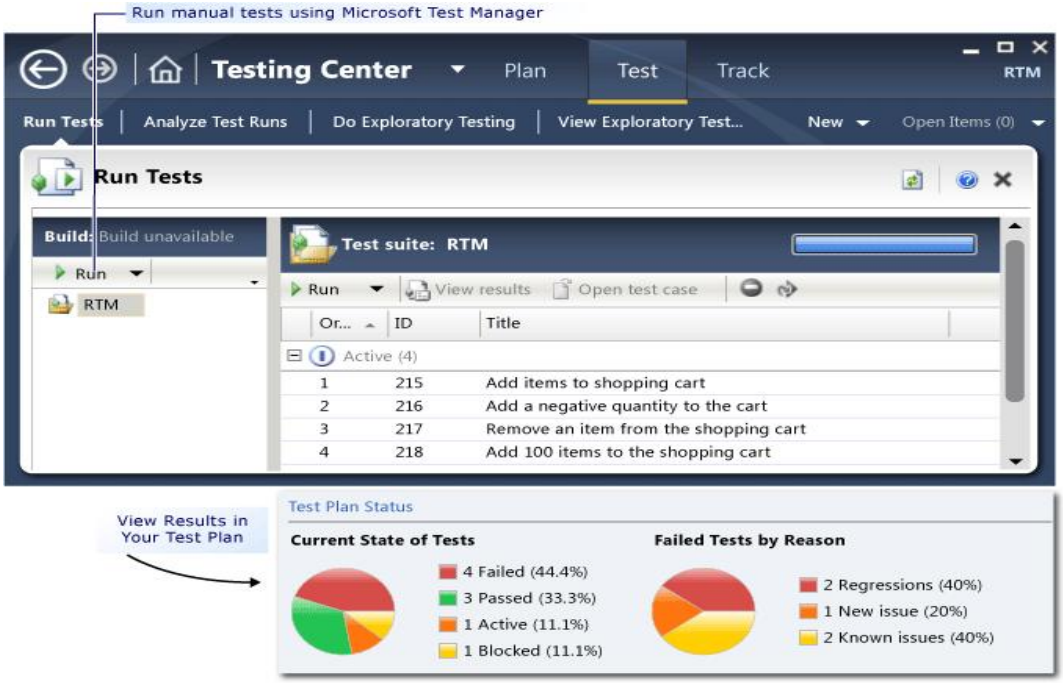
ნახ.14. eAuction Capcha

აღწერილი პრობლემის თავიდან არიდების მრავალი გზა არსებობს, თუმცა აღნიშნული სცენარის რეალიზებისას არსებული Captcha-ს პრობლემა თავიდან არიდებულ იქნა სპეციალურ მომხმარებელზე უფლებების გაწერით(ნულოვანი სიმბოლოების ვალიდურობა - „რეგრესტესტ“ მომხმარებელზე).

Team Foundation Server-ის(TFS) საშუალებით შესაძლებელია ავტომატური build-deploy-test პროცესის გაშვება და პროგრამული „ბილდის“ დროს აპლიკაციის სერვერზე ავტომატური განთავსება-ტესტირება. Vsual Studio მხოლოდ ერთი ბრძანების საშუალებით შესაძლებელია Buil-ის გაშვება, აპლიკაციის განთავსება და ტესტირება, აღწერილ გარემოს ეწოდება BDT. BDT წარმოადგენს Lab მენეჯმენტის ნაწილს, რომელიც აპლიკაციას ანთავსებს Lab მოწყობილობაზე და უშვებს ტესტებს როგორც build პროცესის ნაწილს.

ავტომატური ტესტების გაშვება შესაძლებელია Visual Studio-დან, თუმცა ტესტირების შედეგი იქმნება Microsoft Test Manager-ში(MTM). MTM-ის გამოყენებით შესაძლებელია გავუშვათ manual ტესტები, exploratory ტესტი და ავტომატური ტესტი. ნებისმიერი ტიპის ტესტის შესრულებისას, პროცესის რეპორტი ინახება Test Manager-ში.

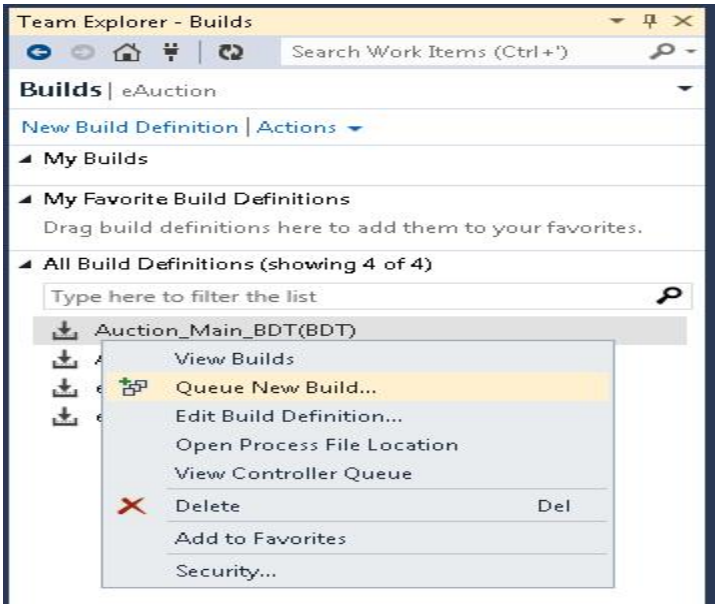
Manual და exploratory ტესტების გასაშვებად საჭიროა MTM-ში გვქონდეს ტესტირების გეგმა, მოცემულ ნახაზზე ილუსტრირებულია ხელით ტესტირება და ტესტირების შედეგი, იხ. ნახ.15.



ნახ.15. MTM: ტესტირების საშედეგო ფაილი

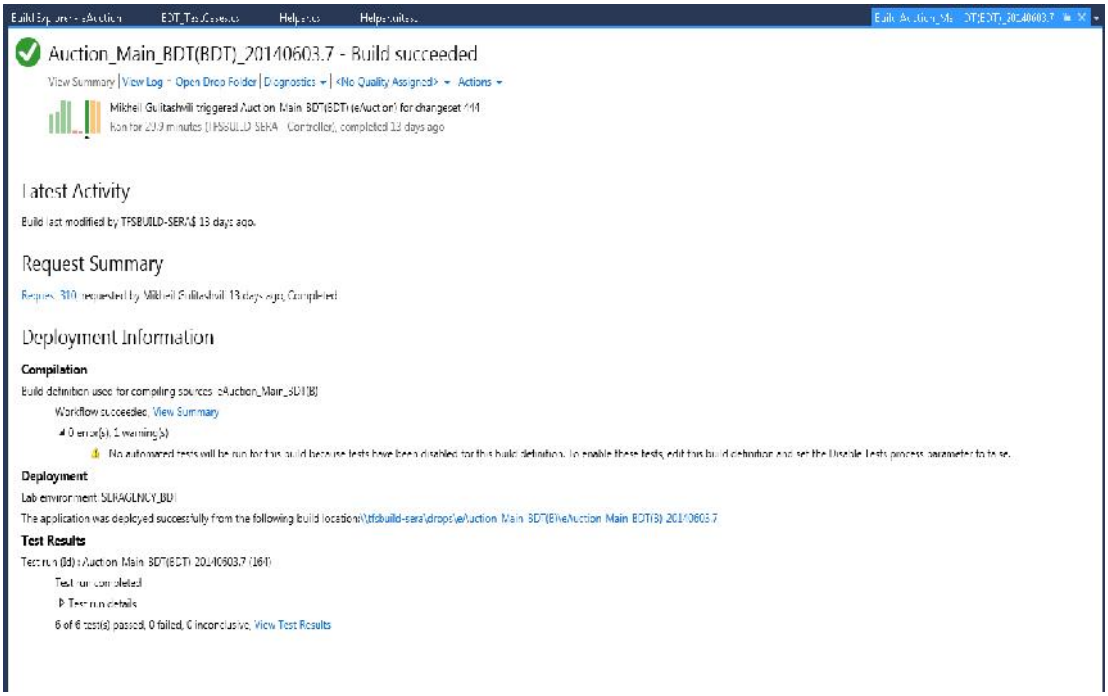
Visual Studio-ს ან MTM-ის საშუალებით შესაძლებელია შემდეგი ტიპის ავტომატური ტესტების გაშვება: Unit tests; Coded UI tests; Load tests.

თუ ზევით განხილულ ამოცანას დავუბრუნდებით და განხილულ eAuction სისტემას გადავიტანთ BDT გარემოში, მაშინ Visual Studio-ს მეშვეობით შექმნილი მომხმარებლის სცენარის, ავტომატური Coded UI ტესტის შესრულებაზე გაშვება და ანალიზი შესაძლებელი იქნება MTM-ის საშუალებით. თუმცა ეს მხოლოდ test ნაწილია, eAuction აპლიკაციის ავტმატური build და deploy შესაძლებელია მხოლოდ TFS-ის საშუალებით, იხ. ნახ.16.

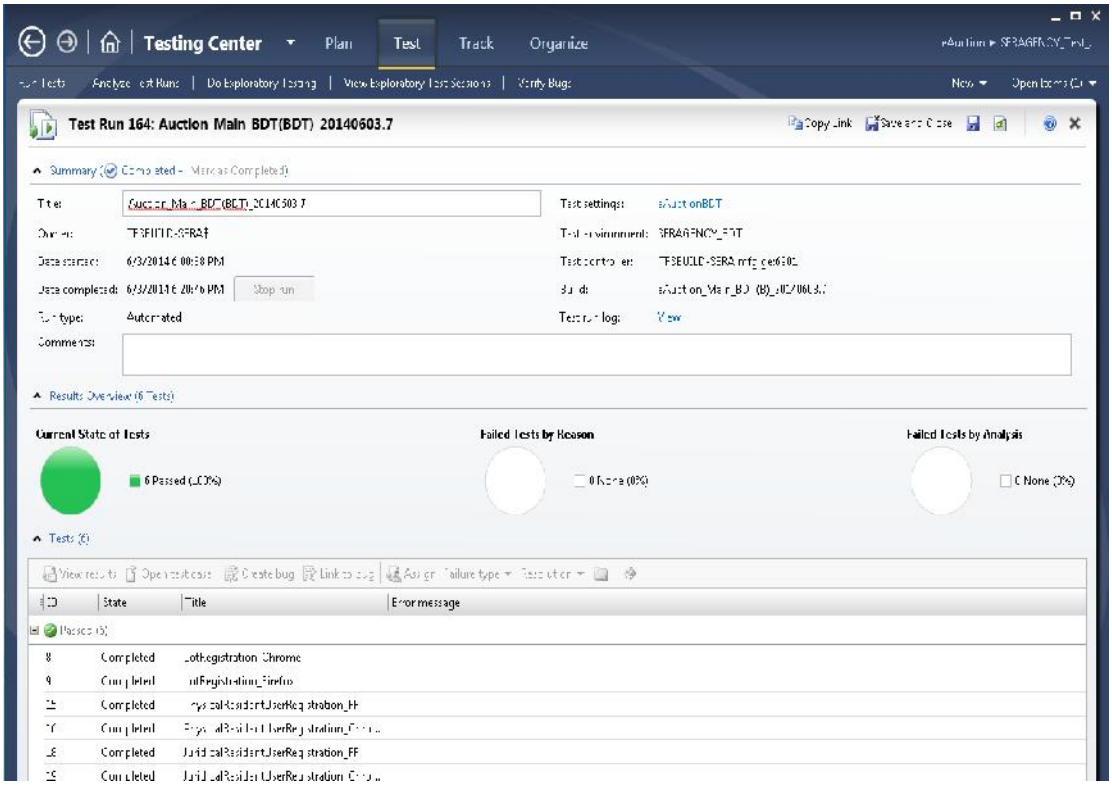


ნახ.16. TFS: BDT პროცესის გაშვება

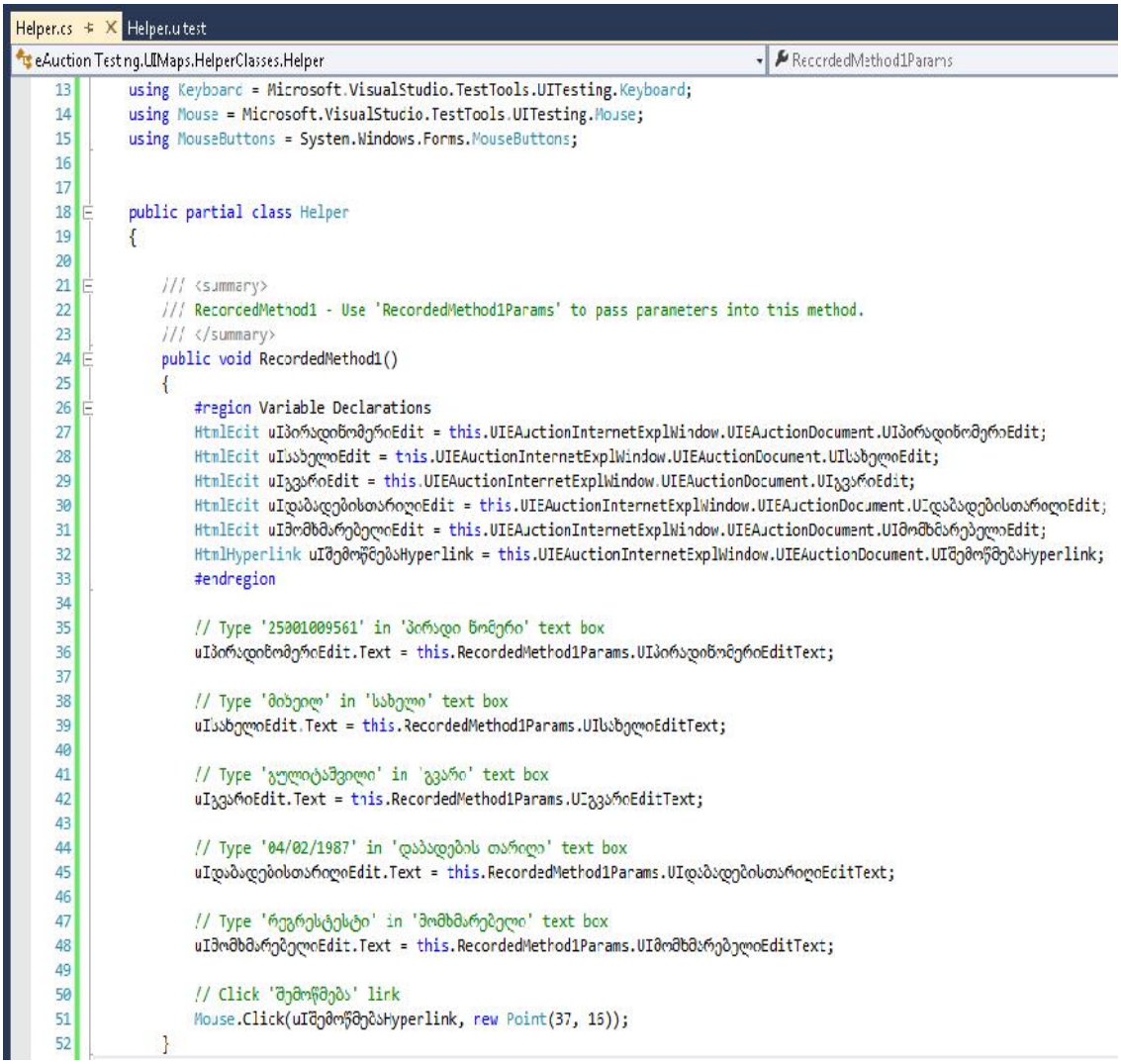
Queqe New Build ბრძანებით გაეშვება eAuction პროექტის ავტომატური Build, Deploy და Test. თუ Build და Deploy პროცესები წარმატებით დასრულდება, ბოლო ეტაპზე გაეშვება ტესტი, მომხმარებლის რეგისტრაციის ავტომატური სცენარი. BDT-ის შედეგის ნახვა შესაძლებელია როგორც Visual Studio-დან (ნახ.17.), ასევე Test Manager-დან (ნახ.18).



ნახ.17. Visual Studio: BDT-ის შედეგი



ნახ.18. Test Manager: BDT-ის შედეგი



ნახ.19. Coded UI: გენერირებული

მესამე თავში გადმოცემულია ბიზნეს–აპლიკაციის, კერძოდ ელექტრონული აუქციონის სისტემის ტესტირების ავტომატიზაციის საკითხები Microsoft Test Manager, Visual Studio 2012, Selenium თანამედროვე პროგრამული ინსტრუმენტით. შემოთავაზებულია Java-ს მატესტირებელი „ფრეიმვორკები“ – JUnit და TestNG. წარმოდგენილია ავტომატური ტესტირების შედეგების აღწერა, რეპორტ ფაილების გენერირება. ტექნოლოგია Selenium Grid-ის მაგალითზე განხილულია პარალელური ავტომატური ტესტირება და მისი აქტუალობა შესრულების დროის თვალსაზრისით. წარმოდგენილია Desktop აპლიკაციების მატესტირებელი ინსტრუმენტი AutoIT და მისი სკრიპტინგ ენა, დემონსტრირებულია ავტომატური სცენარები პრაქტიკული მაგალითებით.

მიმდინარე პროექტში ავტომატური ტესტირების პროცესის დანერგვა განაპირობებს მის წარმატებას, ხარისხს. მიმდინარე სადაველოპმენტო პროექტის ვერსიის განახლებამ შესაძლობა გამოიწვიოს, უკვე არსებული და დამოწმებული ფუნქციონალის გაფუჭება. აღწერილი და დანერგილი BDT

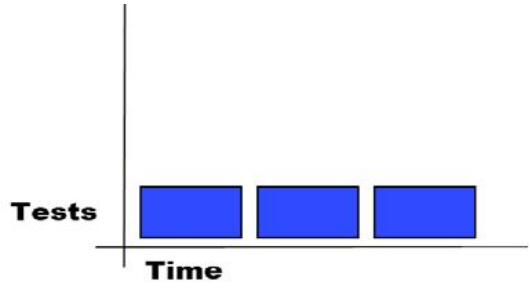
სისტემის საშუალებით მიმდინარეობს პროექტის შესრულებაზე ავტომატურად გაშვება, სერვერზე ავტომატური განთავსება და ავტომატური ტესტირება. ტესტის ჩავარდნის შემთხვევაში პროგრამისტი აანალიზებს შეცდომას, ასწორებს მას და კვლავ თავიდან უშვებს BDT პროცესს და ეს გრძელდება მანამ, სანამ ტესტები არ გამწვანდება. ამრიგად, შეცდომების პოვნა ხდება მათი შექმნიდან უმოკლეს დროში. შესაბამისად, ბაგის მიერ გამოწვეული ზიანი უმნიშვნელოა, განსხვავებით იმ პროგრამული ხარვეზებისა, რომელსაც იჭერს საბოლოო მომხმარებელი.

მოცემულ დისერტაციაში წარმოდგენილია ფინანსთა სამინისტროს ელექტრონული eAuction სისტემის ავტომატური ტესტირების სცენარი „მომხმარებლის რეგისტრაცია“, „ლოტის რეგისტრაცია“, და სხვ. აღწერილი სცენარები დანერგილია და ფართო გამოყენებაშია ფინანსთა სამინისტროს ანალიტიკურ დეპარტამენტში. ანალოგიურად, ჩემს მიერ დაწერილია SilverLight აპლიკაციების მატესტირებელი სცენარები, Unit ავტომატური ტესტები. მაშასადამე, დისერტაციაში აღწერილი და რეალიზებული ავტომატური ტესტები გამოყენებადია მაღალ საპასუხისმგებლო ბიზნეს-აპლიკაციებში.

Selenium Grid არის ავტომატური ტესტების გასაშვები ინსტრუმენტი, რომლის საშუალებით შესაძლებელია ტესტების გაშვება განსხვავებულ მანქანებზე, სხვადასხვა ბრაუზერებზე, სხვადასხვა ოპერაციულ სესტემებზე პარალელურად. ანუ იგი იძლევა დროის ერთიდაიგივე მომენტში მრავალი ტესტის გაშვების შესაძლებლობას სხვადასხვა მანქანაზე, სხვადასხვა ბრაუზერზე და ოპერაციულ სისტემაზე. აგრეთვე Grid–ს შეუძლია განაწილებული ტესტების გაშვება.

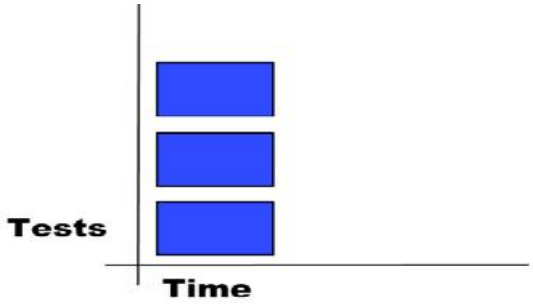
Selenium Grid გამოიყენება ტესტირების შესრულების დროის დასაჩქარებლად. იგი შესრულების სიჩქარეს ზრდის რამოდენიმე მანქანის გამოყენებით და მათზე პარალელური ტესტების რეალიზებით. მაგალითად, თუ გვაქვს შესასრულებელი 100 ტესტი, და ჩვენ დავაყენებთ Selenium Grid–ს 4 სხვადასხვა მანქანაზე, მოცემული ტესების გასაშვებად დაგვჭირდება 4 ჯერ ნაკლები დრო ვიდრე ტესტების ერთ მანქანაზე მიმდევრობით გაშვებისას. დიდი ტესტ–სცენარებისთვის და ხანგრძლივი ტესტებისთვის, როგორიცაა მაგალითად ბევრი მონაცემების ვალიდაცია, პარალელური ტესტირება შეიძლება იყოს მნიშვნელოვანი, დროის დაზოგვის თვალსაზრისით.

ტრადიციულად Selenium ავტომატური ტესტები სრულდებიან იმ თანმიმდევრობით როგორ მიმდევრობასაც მატესტირებელი ფრეიმვორკი (JUnit, TestNG) განსაზღვრავს(ნახ.20).



ნახ.20. მიმდევრობითი ტესტების დროის დიაგრამა

თუ განვიხილავთ ტესტების შესრულებას პარალელურ რეჟიმში, ეს ნიშნავს ტესტების შესრულების პროცესის დაჩქარებას დაახლოებით n–ჯერ, სადაც n არის ბრაუზერების რაოდენობა. პარალელური პროცესის საილუსტრაციოდ იხილეთ ნახ.21



ნახ.21. პარალელური ტესტების დროის დიაგრამა

AutoIt ორიენტირებულია Windows GUI(Graphical User Interface)–ის ავტომატიზაციისთვის. იგი Microsoft Windows-თვის წარმოადგენს **უფასო** ავტომატიზაციის ენას. AutoIt გამოიყენება ღილაკებზე დაჭერის, მაუსის გადაადგილების, ფანჯრების მართვის იმიტირების ავტომატიზაციისთვის. იგი არის ძალიან პატარა პროგრამული უზრუნველყოფა, რომელიც მუშაობს მხოლოდ Windows ოპერაციული სისტემის ყველა ვერსიაზე.

ისტორიულად AutoIt შეიქმნა პერსონალური კომპიუტერის ავტომატიზაციისთვის, რომლის მიზანს წარმოადგენდა ათასობით კომპიუტერის მართვა. დროთა განმავლობაში იგი გახდა უფრო მძლავრი, დამოუკიდებელი ენა, რომელსაც შეუძლია რთული ფუნქციების შესრულება, მომხმარებლის მოქმედებების იმიტირება, ციკლები, მასივები და ყველა იმ ფუნქციის შესრულება რასაც "ვეტერანი" სკრიპტინგ–ენები აკეთებს.

AutoIt–ის აქვს Basic–ის მსგავსი სინტაქსი, რაც იმის მანიშნებელია რომ ადამიანებს რომლებსაც როდესმე დაუწერიათ სკრიპტი, ან გამოუყენებიათ მაღალი დონის პროგრამირების ენა, ადვილად შეუძლიათ შეისწავლონ მისი სინტაქსი. იმის მიუხედავად რომ მან დაიწყო ცხოვრება როგორც მარტივმა ავტომატურმა პროგრამამ, ამჟამად მას აქვს ისეთი

რთული ფუნქციები და მახასიათებლები როგორც ძირითად სკრიპტინგ ენებს. ქვემოდ მოცემულია AutoIT-ზე დაწერილი ავტომატური სცენარის ფრაგმენტი, AutoIT სკრიპტი:

```
Run("notepad.exe")
WinWaitActive("Untitled - Notepad")
Send("Hey GTU!")
WinClose("Untitled - Notepad")
WinWaitActive("Notepad", "Save")
Send("!s")
WinWaitActive("Save As", "*.txt")
Send("AutomateNotepad")
Send("!s")
WinWaitActive("Confirm Save As", "Yes")
Send("!y")
Exit
```

ნახ.21. ავტომატური სცენარის სკრიპტი

დასკვნა:

სადისერტაციო თემის ფარგლებში ჩატარებული საპროექტო–კვლევითი სამუშაოების შედეგების საფუძველზე შესაძლებელია შემდეგი დასკვნების გაკეთება:

1. სადისერტაციო ნაშრომში „პროგრამული სისტემების ავტომატური ტესტირება“, განხილულია გამოყენებითი სფეროს მართვის ავტომატიზებული სისტემების ტესტირების, ვერიფიკაციისა და ვალიდაციის ამოცანები ობიექტ-ორიენტირებული მიდგომის საფუძველზე. ყურადღება გამახვილებულია პროგრამული პროექტების ავტომატური ტესტირების საკითხებზე, ობიექტზე ორიენტირებული თანამედროვე ტექნოლოგიების საშუალებით. შემოთავაზებულია პროგრამული სისტემების ავტომატური ტესტირების მოდელები, მათი კლასიფიკაცია და რეალიზაციის გზები. შესაბამისი თვალსაზრისით რეალიზებულია პრაქტიკული ექსპერიმენტების შედეგები MS Visual Studio 2012/Selenium RC/WebDriver/AutoIT გარემოში.

2. განხილულ იქნა პროგრამული უზრუნველყოფის, Web აპლიკაციების ავტომატური ტესტირება Selenium ტექნოლოგიით. ავტომატური ტესტირების გამოყენების სფეროები და მისი შედარება ხელოვნურ ტესტირებასთან, უპირატესობები და ნაკლოვანებები. Selenium ტექნოლოგიის ზოგადი აღწერა, მისი შექმნის ისტორია და განვითარება. ფართოდ მიმოხილულია Selenium ტექნოლოგიის ძირითადი შემადგენელი მატესტირებელი ინსტრუმენტები ამოცანების პრაქტიკული რეალიზებით.

3. პროგრამული სისტემების სასიცოცხლო ციკლის მენეჯმენტის საკითხი, კერძოდ მისი ტესტირების ეტაპი. განხილულია პროგრამული

უზრუნველყოფის ტესტირების სისტემების კლასიფიკაცია, მათ შორის Desktop და Web აპლიკაციების ტესტირების ავტომატიზაციის მეთოდოლოგიები, აგრეთვე მათი გამოყენების სფეროები თანმედროვე ინფორმაციულ ტექნოლოგიებში.

4. პროგრამული უზრუნველყოფის ტესტირების ძირითადი ტიპები და მათი გამოყენების სფეროები. რომელი ტიპის ტესტირება გამოიყენება აპლიკაციის შექმნის სხვადასხვა ეტაპზე და ვის მიერ სრულდება აღწერილი ტესტირების ტიპი. შემოთავაზებულ იქნა პროგრამული ტესტირების ძირითადი ცნებების განმარტებანი, სცენარში შემავალი ტესტ-ქეისების ტიპები და მათი პრაქტიკული რეალიზაცია Selenium ტექნოლოგიით.

5. შემუშავებულია Web სისტემების ტესტირების პროცესის ავტომატიზაციის, დამუშავების და რეალიზაციის საკითხები თანამედრვე Open Source და კომერციული ინსტრუმენტებით. ტესტირების ავტომატიზაცია ხორციელდება Selenium მატესტირებელი ინსტრუმენტებით, სცენარების რეალიზაცია კი HTML/Java/C# ტექნოლოგიებით. განხილულია ტესტების მიმდევრობითი და პარალელური შესრულების შეფასებები, ტესტირების რეპორტი. პარალელური ტესტირების შედარება მიმდევრობით ტესტირებასთან, სკრიპტული ენით დაწერილი ტესტის შედარება პროგრამული ენით რეალიზებულ ტესტთან. განხილულია მატესტირებელი ტექნოლოგიების სხვადასხვა ბრაუზერებთან, ოპერაციულ სისტემებთან და პროგრამულ ენებთან თავსებადობის საკითხები. ავტომატური ტესტირების შედარება ხელით ტესტირებასთან და მათი გამოყენების სფეროები.

6. ნაშრომში განხილულია Microsoft Visual Studio 2012 ALM ტექნოლოგია, მატესტირებელი BDT პლატფორმა. განხილულია ფინანსთა სამინისტროს სახელმწიფო ელექტრონული აუქციონის - eAuction დეველოპმენტის ავტომატიზაციის პროცესი, თუ როგორ მიმდინარეობს ბიზნეს აპლიკაციის ავტომატური Build, Deploy და Test (BDT). აღნიშნული მაგალითის საფუძველზე განხილულია Silverlight აპლიკაციის ტესტირების რამოდენიმე ტიპი: Unit Testing, Regression Testing და Functional Testing. აღწერილია Microsoft-ის ტესტირების პლატფორმა Microsoft Test Manager (MTM) და მოკლედ მიმოხილულია ავტომატური ტესტირებისთვის საჭირო ტექნიკური მოწყობილობები. შესაბამისი თვალსაზრისით რეალიზებული პრაქტიკული ექსპერიმენტების შედეგები განხორციელებულია MS Visual Studio 2012/MTM/Test Agent/Test Controller/MS SQL Server ინსტრუმენტებით.

გამოქვეყნებული ლიტერატურა:

1. გ.სურგულაძე, **მ.გულიტაშვილი,** ი.კაკულია, გ.ჩერქეზიშვილი, ი.ჯავახიშვილი. პროგრამული სისტემების სასიცოცხლო ციკლის პროცესის მოდელირება უნივერსალური და ექსტრემალური პროგრამირების პრინციპების კომპრომისული გადაწყვეტით. სტუ-ს შრ.კრ. „მართვის ავტომატიზებული სისტემები“, N 1(8). **2010**, გვ.63-70

2. **მ. გულიტაშვილი,** გ. სურგულაძე, გ. ჩერქეზიშვილი. Web აპლიკაციების დამუშავების პროცესის მოდელირება UML/2 ტექნოლოგიით. სტუ-ს შრ.კრ., „მართვის ავტომატიზებული სისტემები“, N 1(10), **2011**. გვ.180-184

3. **მ. გულიტაშვილი,** გ. სურგულაძე. Web-აპლიკაციების ავტომატური ტესტირება. სტუ-ს შრ.კრ. „მართვის ავტომატიზებული სისტემები“, N 1(12), **2012**. გვ.41-45

4. **მ. გულიტაშვილი.** Web აპლიკაციების ავტომატური ტესტირება SELENIUM ტექნოლოგია. სტუ-ს შრ.კრ. „მართვის ავტომატიზებული სისტემები“,
N 2(13), **2012**. გვ.199-202

5. **მ. გულიტაშვილი,** გ. სურგულაძე, გ. ჩერქეზიშვილი. პროგრამული უზრუნველყოფის ტესტირების ტიპები და სცენარები. სტუ-ს შრ.კრ. „მართვის ავტომატიზებული სისტემები“, N 1(14), **2013**. გვ.95-99

6. გ. სურგულაძე, **მ. გულიტაშვილი,** ა. რამიშვილი. Web სისტემების ტესტირების ავტომატიზაცია Open Source ტექნოლოგიების გამოყენებით. ინსო–2013, VI საერთაშორისო სამეცნიერო **კონფერენცია** „ინტერნეტი და საზოგადოება“. ქუთაისი. **2013**. გვ. 15-19

7. **მ. გულიტაშვილი,** გ. სურგულაძე, ბ. ჩიხრაძე. Web აპლიკაციების დამუშავების პროცესის მოდელირება UML/2 ტექნოლოგიით. საერთაშორისო სამეცნიერო **კონფერ. მოხსენებათა თეზისები:** „მართვის ავტომატიზებული სისტემები და თანამედროვე საინფორმაციო ტექნოლოგიები“. ISBN 978-9941-14-937-5. სტუ, თბ., **2010**. გვ.168-169

8. ნ.კივილაძე, გ.სურგულაძე, **მ.გულიტაშვილი.** Microsoft-ის ტექნოლოგიების ანალიზი და განვითარების ტენდენციები საინფორმაციო სისტემებისთვის. სტუ-ს შრ.კრ. „მართვის ავტომატიზებული სისტემები“, N 1(17), **2014**. გვ.199-202.

ABSTRACT

In the represented dissertation "Organizational management IT - consulting models and methods processing and analysis for integration of software systems", described software life cycle process issues, for example automated testing, verifications and validations problems by Object Oriented base methodology. Direct one's attention to systems integration application automated testing models, algorithmically scheme and scenario building, software systems automatic testing issues, by used of modern object oriented technologies. Software systems automatic testing models is offered, their classification and realization ways. According point of view is realized practical experiments results in MS Visual Studio 2012/Selenium RC/WebDriver/AutoIT environment.

Dissertation involves of software Web applications automatic testing by Selenium technology, Automatic testing usages and its comparison to manual testing, advantages and disadvantages. General description of Selenium technology is represented, discussed creation history and development. By solving practical problems widely reviewed Selenium technology's main testing tools.

Management of software systems life cycle issues is offered, application testing steps. In this document is discussed about software testing systems classification, among Desktop and Web applications testing automatisation methodologies, also their usages in modern information technology.

There is carried out software testing main types and their usages issues. Which type of software testing is used during application developing process and by who is executor of described type of testing? Described program testing main term definitions, test-cases types included in test scenario and their practical realization by Selenium technology.

Work out Web systems testing process automatisation and realization aspects by modern Open Source and commercial tools. Test automatisation is implemented by Selenium testing tools; Scenario realization is done by HTML/Java/C# technologies. There is described tests consecutive and parallel execution estimation, testing results or reports. Parallel tests is compared to consecutive testing, test created by scripting language is compared to test realized by programming language. Also here is carried out the compatibility of testing technologies to different internet browsers, operational systems and programming languages. Automatic testing is compared to manual testing and discussed their usage fields.

In the dissertation described Microsoft Visual Studio 2012 ALM technology, the testing BDT platform. Considered Ministry of Financial domestic debt and investment projects managing program (eDMS) developing automatisation process,

how is processing business application automatic Build, Deploy and Test (BDT). By considered example is a represented Silverlight application several testing types: Unit Testing, Regression Testing and Functional Testing. Described Microsoft testing platform – Microsoft Test Manager (MTM) and generally review the necessary technical hardware for automatic testing. Results of experiment realized by using tools: MS Visual Studio 2012/MTM/Test Agent/Test Controller/MS SQL Server.

There is carried out practical Coded UI project of automation testing and test scenario implementation by Visual Studio, Microsoft Test Manager Tools. By given task realization is illustrated the resistances during automatization testing, like captcha picture, problem coming from regression running – user already exists in Database. Work out practical solving ways of described problems. Widely reviewed technical properties of Coded UI testing: structure, contained .uitest file, map of elements, scenario modification by Object-Oriented methodology, source code automated generalization and so on. Illustrated recorded scenario snippet and compared to Selenium technology. Running and debug of realized scenario by Test Explorer Tool result of automatic testing in Visual Studio IDE. By realized scenario example is reviewed Build-Deploy-Test workflow of Team Foundation Server (TFS). Running BDT workflow from TFS and result analayzing. Microsoft Test Manager Tool is described, it's supporting of different type of automatic test: Unit Test, Load Test, Coded UI test. Reviewed it's supporting to Manual and Exploratory type of testing.