პროგრამული უზრუნველყოფის ტესტირების ტიპები და სცენარები

მიხეილ გულიტაშვილი, გია სურგულაძე, გიორგი ჩერქეზიშვილი
საქართველოს ტექნიკური უნივერსიტეტი

რეზიუმე

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

საკვანძო სიტყვები: პროგრამული უზრუნველყოფა. ტესტირება. Selenium.

1. შესავალი

თანამედროვე ინფორმაციულ ტექნოლოგიებში გავრცელებულია მრავალი ტიპის აპლიკაციები, რომლებიც უზრუნველყოფენ მომხმარებელთა სხვადასხვა მოთხოვნილებების დაკმაყოფილებას. გავრცელებულია ისეთი გლობალური აპლიკაციები როგორიცაა ERP(Enterprise Resource Planning), E-Commerce სისტემები, BPM(Business Process Management), HR (Human Resources), და ა. შ. მსგავსი ტიპის აპლიკაციებს იყენებს დიდი კორპორაციები, სახელმწოფო სტრუქტურები, რათა მოახდინოს თავისი საქმიანობის ადვილად მართვა. ასევე არსებობს ე.წ. Desktop აპლიკაციები (სამაგიდო ანუ Windows აპლიკაციები), რომლებიც ადამიანის მიერ გამოიყენებიან ისეთი მოთხოვნილებების დასაკმაყოფილებლად როგორიცაა: გამოთვლები, თამაშები, ტრეინინგები, გართობა და სხვ. აგრეთვე არსებობს სხვადასხვა პლატფორმები (ე.წ. ოპერაციული სისტემები), რომლებიც კომპიუტერული მოწყობილობების რესურსების გამოყენებით უზრუნველყოფს პროგრამული აპლიკაციების მართვას. სხვადასხვა ტიპის კომპიუტერულ აპლიკაციებს აქვს სხვადასხვა არქიტექტურა, შესაბამისად განსხვავდება მათი შექმნის პროცესიც. მაშასადამე, იმისთვის რომ ვიზრუნოთ სხვადასხვა ტიპის პროგრამული უზრუნველყოფის ხარისხზე, ამისათვის არსებობს მრავალი ტიპის პროგრამული ტესტირება [1].

2. ძირითადი ნაწილი

განვიხილოთ პროგრამული უზრუნველყოფის ტესტირების ძირითადი ტიპები:

- **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.1). აპლიკაციის შიგა პროცესები უგულებელყოფილია.

საბოლოოდ, შეგვიძლია დავასკვნათ, რომ "შავი ყუთის" ტესტირება არის ტესტირების ტიპი, რომლის დროსაც ხდება პროგრამის ფუნქციონალის შემოწმება შიგა სტრუქტურის გამოკვლევის გარეშე [1].

ნახ.1. 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 პროგრმული კოდის ცოდნაზე, კომპონენტებზე და მათ შორის კავშირებზე, ციკლებზე, და სხვ [4].

- **Unit testing** (ერთეულოვანი ტესტირება) – როგორც დასახელებიდან ჩანს, "Unit" ტესტირება არის მცირე ზომის ტესტირების მეთოდი, რომლის დროსაც მიმდინარეობს მარტივი მოდულების, ფუნქციების ტესტირება. აღნიშნული ტიპის ტესტირებას აკეთებს პროგრამისტი და არა ტესტერი, რადგან იგი მოითხოვს პროგრამის შიგა დიზაინის, კოდის დეტალურ ცოდნას. ზოგადად, მსგავსი ტიპის ტესტირებას მიმართავენ მაშინ, როდესაც აპლიკაციის არქიტექტურის დიზაინი დასრულებულია [1].

- **Incremental testing** (ზრდადი ტესტირება) – ტესტირების მეთოდი, რომლის დროსაც პირველ ეტაპზე მიმდინარეობს ქვესისტემების დამოუკიდებელი ტესტირება, შემდგომ ეტაპზე ტესტირების პროცესი გადადის გაერთიანებულ ქვესისტემებზე და გრძელდება მანამ, სანამ არ მივიღებთ მთლიან სისტემას.

- **Functional testing** (ფუნქციონალური ტესტირება) – "შავი ყუთის" ტიპის ტესტირება, რომლის დროსაც ტესტირდება პროგრამის ფუნქციონალი შესასვლელზე მიცემული მონაცემებით და მიღებული შედეგებით. მის დროს განიხილება პროგრამის შიგა სტრუქტურაც.

ფუნქციონალური ტესტირება მოიცავს პროგრამის სამომხმარებლო ინტერფეისის, მონაცემთა ბაზის და პროგრამული უსაფრთხოების ტესტირებას. ფუნქციონალური ტესტირება შეიძლება განხორციელდეს ხელოვნურად ან ავტომატურად.

- **System testing** (სისტემური ტესტირება) – დასრულებული, მთლიანი კომპიუტერული პროდუქტის ტესტირების მეთოდი. ზოგადად სისტემური ტესტირება მოიცავს სხვადასხვა ტესტების სერიას, რომელთაც აქვს ერთადერთი მიზანი: დაატესტირონ კომპიუტერზე ბაზირებული მთლიანი სისტემა. აპლიკაცია არის კომპიუტერზე ბაზირებული სისტემის შემადგენელი ერთი–ერთი პატარა ელემენტი.

- **Regression testing** (რეგრესიული ტესტირება) – მისი მიზანია დაადასტუროს, რომ ახალმა პროგრამულმა ცვლილებამ არ გამოიწვია უკვე არსებული და შემოწმებული ფუნქციონალის "გაფუჭება". "Regression" ტესტირების დროს მიმდინარეობს უკვე გაშვებული ტესტების ხელახალი გაშვება. აღნიშნული ტესტირების ტიპი უზრუნველყოფს, რომ ახალ პროგრამულ ცვლილებას არ აქვს

გვერდითი მოვლენები არსებულ ფუნქციონალზე. "Regression" ტესტების გაშვების აუცილებლობა დგება მაშინ, როდესაც ხდება ქვემოთ მითითებული ერთ–ერთი მოვლენა:

- ✓ იცვლება მომხმარებლის მოთხოვნები, მაშასადამე იცვლება კოდი მოთხოვნების შესაბამისად;
- ✓ პროგრამას ემატება ახალი მახასიათებელი;
- ✓ ფიქსირდება პროგრამული დეფექტი.

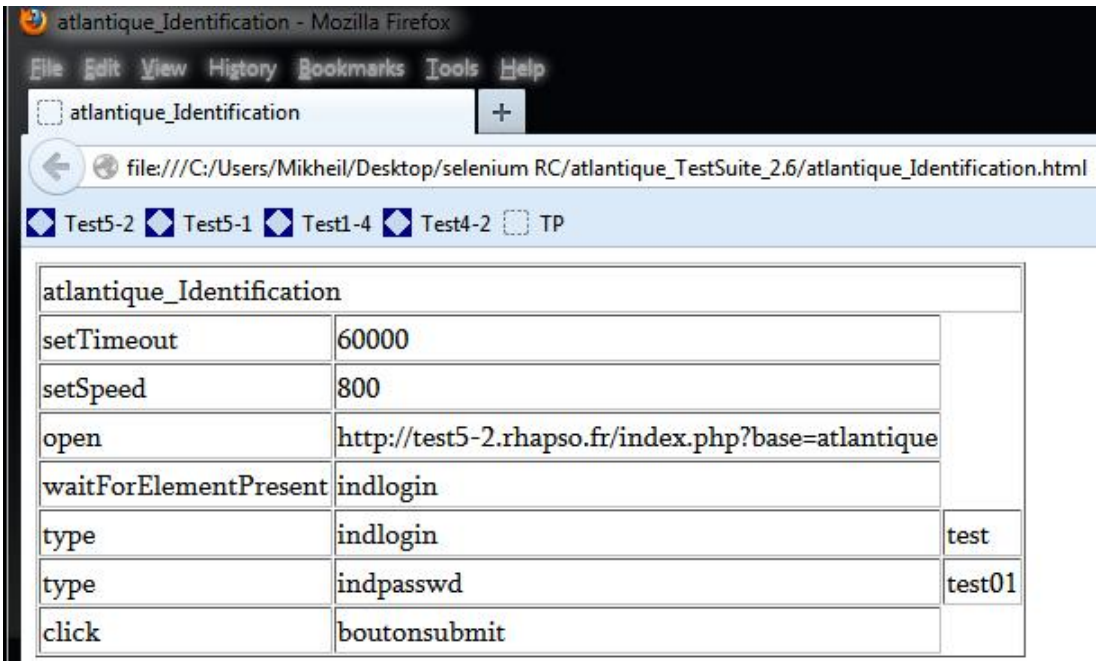
მსგავსი ტიპის ტესტირებისას გამოყენებადია ავტომატური მატესტირებელი ინსტრუმენტები, მაგ. Selenium, რადგან ყოველი პროგრამული ცვლილებისას მიმდინარეობს ერთიდაიგივე ტესტების მრავალჯერადი გაშვება [1,5].

- **Acceptance testing** (თანხმობის ტესტირება) – კლიენტის მიერ შესრულებული ტესტირების პროცესი, რომლის დროსაც ხდება დასრულებული სისტემის შემოწმება, ეთანადება თუ არა სისტემა მყიდველის მოთხოვნებს. იგი არის ტესტირების ერთ–ერთი ბოლო ფაზა, მანამ, სანამ მოხდება აპლიკაციის კომერციულ წარმოებაში გაშვება.
- **Load testing** (დატვირთვით ტესტირება)– აპლიკაციის ტესტირება რთულად დასამუშავებელი მონაცემებით, მაგალითად Web საიტის დატვირთვით ტესტირება, მისი მიზანია განისაზღვროს თუ რა წერტილში ქვეითდება, ნელდება ან ეკიდება სისტემის მუშაობა [2].
- **Stress testing** (სტრესული ტესტირება) – ტესტირების მეთოდი, რომლის დროსაც ხორციელდება სისტემის "სტრესული" რეჟიმით მუშაობა, რათა შემოწმდეს როდის ან როგორ ჩავარდება იგი. პროგრამის დასამუშავებელ მონაცემებად განიხილავენ დიდ რიცხვებს, ან მონაცემთა ბაზის რთული მოთხოვნების გაშვებას და სხვ.
- **Usability testing** (გამოყენებადი ტესტირება) – საბოლოო მომხმარებლის მიერ ყველაზე ხშირად გამოყენებადი ფუნქციების ტესტირება. მისი მიზანია გამოყენებად ფუნქციებში გამოააშკარაოს პროგრამული დეფექტები.
- **Security testing** (უსაფრთხოების ტესტირება) – ტესტირების მეთოდი, რომლის დროსაც მოწმდება თუ როგორ არის დაცული სისტემა არასანქცირებული გარე თუ შიგა წვდომისგან, წინასწარგანზრახული დაზიანებისგან და ა.შ. მის დროს ხორციელდება სისტემის ყველა შესაძლო სუსტი წერტილის ტესტირება, რათა არასანქცირებული შეტევისას არ მოხდეს ინფორმაციის წაშლა, ინფორმაციის მოპარვა, პროგრამის მწყობრიდან გამოყვანა და ა.შ. [1]

პროგრამული უზრუნველყოფის ტესტ–ქეისი არის იმ ცვლადების, პირობების, მოქმედებების ერთობლიობა, რომელთა საშუალებით მოწმდება პროგრამული უზრუნველყოფის მუშაობის სისწორე. ტესტ–ქეისების საშუალებით ადგენენ სისტემის მუშაობის ხარისხს.

დაწერილ, შექმნილ ტესტ–ქეისს ეწოდება ტესტ–სკრიპტი. ტესტ–სკრიპტი არის ხელოვნურად ან ავტომატურად ჩაწერილი სკრიპტი, რომელიც ახდენს შესაბამის ტესტ–ქეისში მოცემული ინსტრუქციების შესრულებას [3].

შესაძლებელია დაიწეროს ხელოვნური ტესტ–სკრიპტი, რომელიც შესრულდება ადამიანის მიერ, ან რაიმე ინსტრუმენტის საშუალებით მოხდება ტესტ–ქეისში მოცემული ინსტრუქციების ავტომატიზაცია. მოცემულ სურათზე წარმოდგენილია Selenium-ის ავტომატური ტესტ–სკრიპტი (იხ ნახ.2).

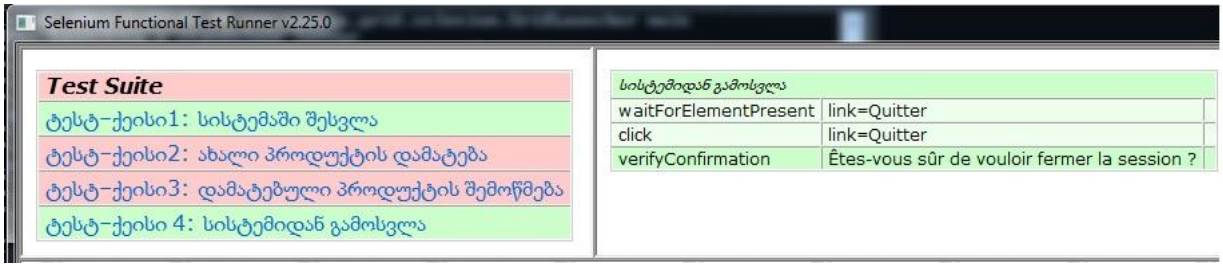


ნახ.2. ტესტ–სკრიპტი

სცენარი არის ტესტ-ქეისების ერთობლიობა, ხოლო ტესტ–სუიტი წარმოადგენს ტესტ–სკრიპტების ერთობლიობას. იმისთვის რომ დატესტირდეს არსებული სისტემის ერთი რაღაც მთლიანი ფუნქცია, ამისთვის ტესტ–ქეისები ერთიანდება სხვადასხვა ჯგუფებად, ტესტ–სუიტებად [3]. მაგალითად, განვიხილოთ ტესტ–სუიტის მაგალითი რომელიც შედგება 4 ტესტ–სკრიპტისგან:

- ტესტ–ქეისი 1: სისტემაში შესვლა
- ტესტ–ქეისი 2: ახალი პროდუქტის დამატება
- ტესტ–ქეისი 3: დამატებული პროდუქტის შემოწმება
- ტესტ–ქეისი 4: სისტემიდან გამოსვლა

განხილულ ტესტ–სუიტში მნიშვნელოვანია ტესტ–ქეისების მიმდევრობა, თითოეული ტესტის წარმატებით დასრულება. მაგალითად წარმოვიდგინოთ რომ ტესტ–სუიტის შესრულებისას არ შესრულდა ახალი პროდუქტის დამატება. ამ შემთხვევაში მომდევნო ტესტი – "დამატებული პროდუქტის შემოწმება" ჩავარდება, რაც ლოგიკურია – თუ ვერ შეიქმნა პროდუქტი, შესაბამისად ვერ მოხდება მისი შემოწმებაც. ამ შემთხვევაში ამბობენ რომ სუიტში შემავალი ტესტ–ქეისები დამოკიდებულნი არიან ერთმანეთზე, ანუ შემდგომი ტესტ–სკრიპტის შესრულება დამოკიდებულია წინა ტესტ–სკრიპტის შესრულების შედეგზე(იხ ნახ.3).



ნახ.3 ტესტ–სუიტი

მწვანე ფერით მოცემულია წარმატებით შესრულებული ტესტ–სკრიპტები, ხოლო წითელ ფერში წარმოდგენილია ჩავარდნილი ტესტები. მოცემული სურათიდან შეგვიძლია დავასკვნათ, რომ ჩავარდა "ახალი პროდუქტის დამატება" ტესტი, შესაბამისად ჩავარდა მასზე დამოკიდებული მომდევნო ტესტი – "დამატებული პროდუქტის შემოწმება", ხოლო წარმატებით შესრულებულა სისტემაში "შესვლა – გამოსვლის" ტესტები [3].

3. დასკვნა

თანამედროვე ინფორმაციულ ტექნოლოგიებში გავრცელებულია მრავალი ტიპის აპლიკაციები, რომლებიც უზრუნველყოფენ მომხმარებლების სხვადასხვა მოთხოვნილებების დაკმაყოფილებას. იმისთვის რომ ვიზრუნოთ სხვადასხვა ტიპის პროგრამული უზრუნველყოფის ტესტირებაზე, ამისათვის არსებობს მრავალი ტიპის პროგრამული 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. პროგრამული უზრუნველყოფის ხარისხი განისაზღვრება ტესტირებით. აპლიკაციის ხარისხის განმსაზღვრელი კრიტერიუმებია: საიმედოობა, სტაბილურობა, შენარჩუნებადობა. ტესტირება არის სწორედ ამ კრიტერიუმებით პროგრამის ხარისხის განმსაზღვრელი პროცესი.

ლიტერატურა:

1. http://www.guru99.com/types-of-software-testing.html
2. Ashfaque Ahmed, Software testing as a service, 2010
3. David Burns, Selenium 1.0 Testing Tools, Birmingham 2010
4. http://en.wikipedia.org/wiki/Software_testing
5. Web-აპლიკაციის ავტომატური ტესტირების Selenium-ტექნოლოგია. სტუ შრ.კრ. მას №2(13), 2012. 199-202 გვ.

SOFTWARE ENVIRONMENT TESTING TYPES AND SCENARIOS

Gulitashvili Mikheil, Surguladze Gia, Cherkezishvili Giorgi

Georgian Technical University

Summary

In the represented article there is considered main types of software environment and fields of their usage. There is described useful type of testing and staff used in the software development process. Software testing main conceptions, types of test case included in scenario and their practical realization by Selenium technology is represented.

ТИПЫ И СЦЕНАРИЙ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Гулиташвили М., Сургуладзе Г., Черкезишвили Г.

Грузинский технический университет

Резюме

Рассматриваются основные типы тестирования и сфера их использования: описываются типы тестирования, которые используются на разных этапах создании аппликации, и охарактеризовываются проводящие тестирование лица. Предлагаются определения основных понятий программного тестирования, типы тест-кейсов, входящие в сценарии и их практическая реализация на базе технологии Selenium.