Test ve Veri Otomasyon Yönetimi

Canberk ERKMEN[1], Ahmet İZGİ[2], Ersin GÜLAÇTI[3]

Logo Yazılım, Kocaeli, Türkiye

[1]canberk.erkmen@logo.com.tr,
[2]ahmet.izgi@logo.com.tr,
[3]ersin.gulacti@logo.com.tr

Özet. Kaliteli yazılım üretmenin en önemli yapı taşı, testi doğru ve etkili yapabilmektir. Yazılım geliştirme süreçlerinin temel adımlarından biri olan testin önemi gün geçtikçe artmaktadır. Manuel test ile maliyet, zaman, hata kaçırma riski artmaktadır. Bu riskleri minimuma indirgemek için yapılan test otomasyonu; testlerin otomatik şekilde yapılmasını sağlayarak kullanıcının manuel test etme zorunluluğu ortadan kaldırır. Fakat test otomasyonunun da kendine ait zorlukları bulunmaktadır. Zorluklardan biri her bir otomasyon senaryosu için farklı kod yazılmasıdır. Bu da senaryoyu yazan kullanıcının kodlama yetkinliğine sahip olmasını gerektirir. Ayrıca ürün değişikliklerinde, yapılan otomasyon senaryolarının kodlarının değiştirilmesi, hatta baştan yazılması bile gerekebilmektedir. Bu da senaryoların bakım maliyetini arttırmaktadır. Bir diğer zorluğu da test verisinin yönetilmesidir. Kodlarla iç içe girmiş olan veriler otomasyon senaryolarının bakımını da oldukça zorlaştırır. Yapılan otomasyon senaryolarının çalıştırılması için ortam kurulması, bunların yönetilmesi ve sonuçlarının takip edilebilmesi de test otomasyonunun zorlukları arasında yer alır.

TEDAM (Test and Data Automation Manager), web uygulamaları için uçtan uca test otomasyonunu hazırlayıp, bu otomasyon senaryolarının yönetilmesinde kullanılır. Çalışmamız, test sorumlusunun kodlama bilgisi gerekmeksizin ara yüz ile otomasyon senaryolarını hazırlamasını ve çalıştırabilmesini sağlar. Bununla birlikte her otomasyon senaryosu için farklı bir kod yazmak gerekmediğinden, yazılan kodlar bir standarda bağlanarak üretilmiş olur. Ayrıca otomasyon senaryosu yapılan ekran ve bileşenleri kayıt altına alarak, otomasyon senaryolarının bakım maliyetini azaltır ve verinin akıllı yönetilmesi sağlanır. Son olarak, hazırlanan otomasyon senaryolarını toplu halde, dağıtık olarak, farklı ortamlarda koşturulmasını sağlayan 'İş Yöneticisi' teknolojisi de çalışmamız içerisinde yer almaktadır.

Anahtar Kelimeler: Test Otomasyonu, Test Verisi, Web Otomasyonu, Otomasyon Senaryosu.

Test And Data Automation Management

Canberk ERKMEN[1], Ahmet İZGİ[2], Ersin GÜLAÇTI[3]

Logo Yazılım, Kocaeli, Türkiye

[1]canberk.erkmen@logo.com.tr,
[2]ahmet.izgi@logo.com.tr,
[3]ersin.gulacti@logo.com.tr

Abstract. The most important foundation of producing quality software is to make the test accurate and effective. Software testing which is one of the main steps of the software development process is becoming more important than ever. With manual testing; cost, time and risk of missing a defect is increasing. Test automation which is used to minimize these risks eliminates the need for manual testing by ensuring that the tests are done automatically. Nevertheless, test automation has its difficulties too. One of the difficulties is to write different code for each automation script. This requires that the user who writes the automation script has coding capability. Furthermore, in the case of product changes, the code of the automation scripts may need to be changed or even rewritten. This increases the maintenance cost of the scripts. Another difficulty is the management of the test data which is embedded in the code. This makes it very difficult to maintain automation scripts. It is also a challenge to set up the environment for the execution, management and monitoring of the automation scripts.

TEDAM (Test and Data Automation Manager) is used for organizing end-to-end test automation scripts for web applications and managing them. Our work allows the user who is responsible for test to create and execute test automation scripts via the user interface without the need for coding. Since there is no need to write a different code for each automation script, the codes are generated according to a standard. In addition, our work records the screens and components which are used in automation scripts, reducing the maintenance cost of automation scripts and providing intelligent management of the test data. Finally, 'Job Manager' technology which enables the prepared automation scripts to be run in a distributed manner with different environments collectively, is also part of our work.

Keywords : Test Automation, Test Data, Web Automation, Automation Script

1 Giriş

Yazılım yaşam döngüsünde 1960 yıllarından test otomasyonlarının önem kazanmış olduğu bugüne doğru gelinirken; gerçekleşen efor bölüşümlerinde, yazılım testine ayrılan oranın arttığı görülmektedir [1]. Testlerin manuel olarak yapılması, harcanan süreyi arttırdığı gibi, kapsam olarak da belli bir alanda kısıtlı kalınmasına neden olmaktadır. 2018 yılında sektör çalışanları arasında yapılan uluslararası bir araştırma raporuna göre kuruluşların büyük bir kısmı, yapılan testleri otomatikleştirmek adına test otomasyon faaliyetleri yürütmektedir [2]. Fakat test otomasyonu yapılırken karşılaşılan sorunlar da yapılan test otomasyonun etkinliğini azaltabilmektedir. Örneğin, test yapan uzmanın kodlama gibi teknik alt yapı gerektiren konularda yazılım geliştirici kadar bilgi sahibi olması gerekmektedir. Yazılan kodların bakımının yapılması, otomasyonda kullanılan test verisinin kodla iç içe geçmesi ve yazılan

otomasyon kodlarının çalıştırılacağı ortamların yönetilmesi, sonuçların doğru takip edilebilmesi gibi sorunlar da bulunmaktadır. Logo Yazılım olarak test otomasyonunda karşılaşılan bu tip sorunları aşabilmek adına TEDAM ürünü geliştirilmiştir.

TEDAM; kullanıcı ara yüzü (TEDAMFace), kendisine gelen komutları çalıştıran istemci (TEDAMAgent), sistem yöneticisi (TEDAMEngine), Chrome browser için geliştirilmiş data toplama yardımcısı (TEDAM Chrome Extension), bileşenlerin temelinde bulunan merkez uygulama (TEDAMCore) olmak üzere 5 ana parçadan oluşur. Yapmış olduğumuz bu çalışmada ilk olarak test otomasyonuna neden ihtiyaç duyduğumuz ve test otomasyonu esnasında karşılaşılan zorluklardan bahsedilecektir. Sonrasında da TEDAM ürününün ortaya koyduğu avantajlar ve alt bileşenleri ile çalışma şekli anlatılacaktır.

2 Neden Test Otomasyonu?

Projelerde yazılım testi için ayrılan sürenin, test uzmanları tarafından sadece manuel olarak yapılan testlere ayrılması genel anlamda efor kaybı olarak adlandırılabilir. Çok sayıda test senaryosunun olduğu bir ortamda regresyon testlerinin sürekli manuel olarak koşulması iş yükünü çok fazla arttıracaktır [3]. Test otomasyonu, yazılım test sürelerine doğru şekilde uygulandığında şu faydaları sağlayacaktır:

- Regresyon testlerinin manuel testlere göre daha sık koşulabilmesini sağlar [4].
- Çevik yazılım geliştirme süreçlerine uyumludur [4].
- Test senaryolarının kayıt altında olmasını destekler [4].
- Kullanıcı gözünden kaçabilecek ufak detayları yakalayabilir [5].
- Testlerin derinlik seviyesi ve kapsama alanını arttırır [5].
- Uygulamanın çalışmasına yönelik güveni pekiştirir [6]

3 Test Otomasyonunda Karşılaşılan Sorun ve Zorluklar

Yazılım test otomasyonunun geliştirilmesi uçtan uca düşünüldüğünde kendi içinde bir yaşam döngüsüne sahiptir. Bu yaşam döngüsündeki adımların doğru şekilde işlemesinde test uzmanının ve kullanılan aracın yetkinlikleri ön plana çıkmaktadır. Bu yaşam döngüsündeki temel aksiyonlar şunlardır;

- Test senaryosunun yazılması,
- Kullanılan aracın kodlama diline uygun olarak kodlanması,
- Test ortam ve verilerinin hazırlanması,
- Otomasyon senaryolarının koşularak bakımının yapılması.

Test otomasyonu hazırlanırken test uzmanı olan kişilerin de kodlama yapması gerekmektedir. Kodlama yapmadan senaryo hazırlamaya imkân sağlayan kaydet-oynat prensibi ile çalışan (Selenium vb.) test otomasyon araçları da bulunmaktadır. Fakat bu araçlar da temelde kullanıcının yaptığı eylemleri kod olarak kayıt altına alıp oynatmaktan ileriye gidememektedir. Bu nedenle bakım yapılması gerektiğinde test uzmanının otomasyon senaryosuna ait kodları güncellemesi gerekecektir. Test yapan

kullanıcılara bakıldığında sadece %33 seviyesinde test otomasyon uzmanı olduğunu görüyoruz [7]. Test otomasyonunu sadece kodlama bilen test uzmanlarından ziyade, hiç kodlama bilmeyen test uzmanı, analist veya konu uzmanlarının da yapma gerekliliği, geleneksel otomasyon araçları ile mümkün olmamaktadır. En çok kullanılan test otomasyon aracına bakıldığında da doğrudan kod bilme gereksinimi olan Selenium aracının olduğu görülmektedir [7].

Senaryolarda verilerinin hazırlanan test otomasyon kodlarının için gömülü olması, verilerin belli bir noktadan sonra kontrol altında tutulmasını oldukça zorlaştıracaktır. Bu tarz problemleri çözmek adına ortaya konulmuş olan metotlar bulunmaktadır. Fakat ortaya konulan metotlar, spesifik olarak sadece bu problemi çözmeye odaklandığından resmin bütününde verimli bir iyileştirme ortamı oluşturmamaktadırlar [8].

Testlerin koşulması esnasında, test senaryolarının dağıtık olarak koşturulmasını sağlayan sunucu ve koşu işlemini gerçekleştirecek istemcilerin hazırlanması ve yönetilmesi oldukça zordur. Bu aksiyon genel olarak test otomasyonu kapsamında düşünülmediğinden, birçok test otomasyon aracında bu dağıtım ve takip özellikleri bulunmamaktadır. Testlerin istemcilerde koşulması ve sonuçlarının toplanarak kayıt altına alınması, hatta hangi ortamda, test edilen ürünün hangi versiyon ve data ile test edildiğinin kayıt altında tutulması, takip edilmesi oldukça zordur.

4 Logo Yazılımda Test Otomasyonu

TEDAM ile ortaya konulan otomasyon çözümü öncesinde, tüm vaktini otomasyon çalışmalarına harcayan test uzmanları bulunmasına rağmen, yapılan otomasyonlar ile kod kapsamında, belirlenen ilk hedef olan %50 başarılmıştır. Ayrıca yazılan otomasyon senaryolarının koşulması, bakımının yapılması, dağıtık olarak çalıştırılabilmesi mümkün olmuyordu. Test senaryolarının yeterli düzeyde yazılamaması, nelerin tam olarak test edildiğini anlamayı zorlaştırıyordu.

5 TEDAM'ın Geliştirilmesi

TEDAM; Test and Data Automation Manager kelimelerinin kısaltması ile oluşmuş, test ve veri otomasyon yönetimi anlamına gelen bir kavramdır. Uçtan uca test otomasyonu geliştirilmesi için ortaya konulmuş olan bir yazılım ürünüdür.

5.1 TEDAM ile İlgili Gereksinimlerin Belirlenmesi

Test otomasyonunu verimli ve etkin bir seviyeye çıkarmak için; mevcut çalışmalar incelenmiş, sorun yaşanan ve iyileştirmeye açık olan alttaki konular ortaya çıkarılmıştır:

- Verinin hali hazırdaki otomasyon yöntemlerinde kodun içinde kalması nedeniyle, veriyi kod ile ayıracak bir yapının oluşturulması,
- Test senaryolarının yazıldığı ortam, otomasyon kodlarının hazırlandığı ortam ve otomasyonların çalıştırıldığı ortamların tamamen farklı olmasının önüne geçilerek, tek bir uygulama ile tüm otomasyon eylemlerinin birleştirilmesi,

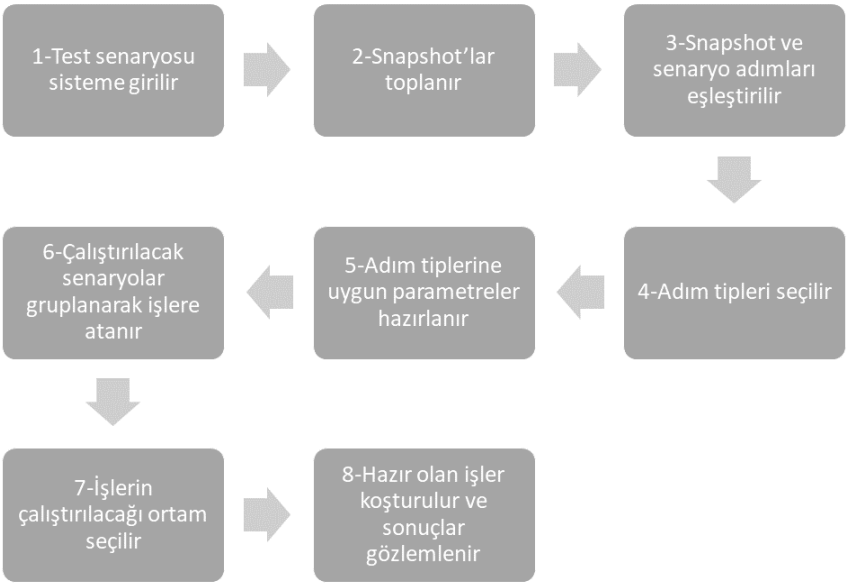
- Test uzmanlarının kodlama yapmak zorunda olması ve otomasyon için yazılan kodların da hataya açık olması nedeniyle, kodun test uzmanı tarafından yazılması yerine sistem tarafından otomatik olarak hazırlanması.

5.2 TEDAM Çalışma Süreçleri

TEDAM'ın işletilmesinde iki tür süreç vardır.

Konfigürasyon Süreci. Test edilecek her yeni proje için bir kere yapılan süreçtir. TEDAMAgent'ların çalışacağı istemcilerin hazırlanması ve tanımlanması, kullanılacak olan komutlar için parametre ve değerlerin tanımlanması, komutların tanımlanması ve TEDAM Chrome Extension üzerinde uygulama ekranlarına bağlı tanımlamaların yapılması adımlarından oluşmaktadır.

Senaryo Giriş Süreci. TEDAM üzerinden senaryo girişinde yapılması gereken adımları kapsayan süreçtir (Şekil 1).



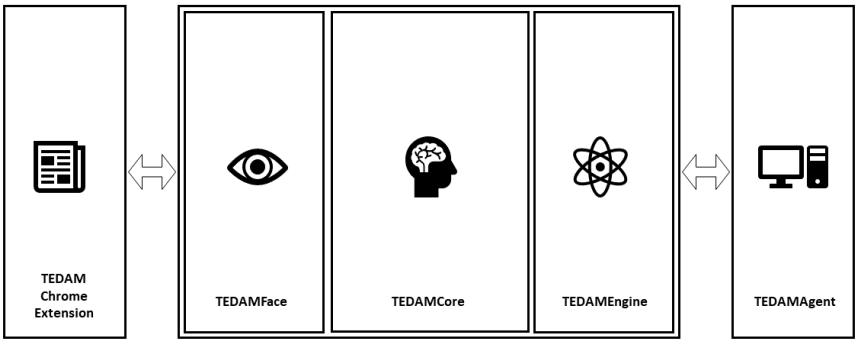
Şekil 1. TEDAM senaryo giriş süreci

Şekil 1'de de gösterilen senaryo giriş süreci; TEDAM kullanıcısının test etmek istediği kısımla ilgili her türlü etkileşimi (yazma, okuma, tıklama) atomik parçalar halinde TEDAMFace aracılığıyla adım adım yazmasıyla başlar (1). Sonrasında TEDAM Chrome Extension kullanılarak her bir ekranın etkileşim yapılan bileşenleri ve giriş yapılan verilerini barındıran XML dosyaları (Snapshot) toplanır (2). Girilmiş olan test adımları ile alınmış olan snapshot dosyaları birebir olarak TEDAMFace üzerinde eşleştirilir (3). TEDAM kullanıcısının girmiş olduğu her bir adım için adım tipini seçmesi gerekmektedir. Bu adım tipleri o anda test adımında etkileşimde olunan uygulama ekranı için; veri girişinde FormFill, bileşen veya dataların kontrol

edilmesinde Verify, bileşenlerin tıklanmasında ButtonClick olarak isimlendirilmektedir (4). Adım tiplerine bağlı olarak açılan TEDAM formlarında, kullanıcı etkileşimde olacağı alanları seçer (5). Kullanıcı çalıştırmak istediği test senaryolarını, anlamlı olarak gruplayarak, test senaryosunun çalışabilir kısmı diye adlandırılan işlere dönüştürür (6). Oluşturulan işler için istemciler, komutlar ve parametreler seçilir (7). Çalışmaya hazır hale getirilmiş olan işler anlık olarak ya da istenen zaman dilimlerinde çalıştırılır (8).

5.3 TEDAM Sistem Mimarisi ve Bileşenleri

TEDAM, şekil 2'de de gösterildiği üzere belirlenen ihtiyaçlara bağlı olarak farklı görev ile sorumlu 5 bileşenden oluşmaktadır.



Şekil 2. TEDAM sistem bileşenleri

TEDAMCore. Bu bileşen sistemin en temel parçası olarak konumlandırılmıştır. Tüm uygulamanın veri tabanı ile olan etkileşimi bu bileşen aracılığıyla sağlanır. Hazırlanmış olan test otomasyon senaryolarının kodlarının üretilmesini sağlar. TEDAMAgent ile TEDAMEngine arasındaki iletişime aracılık etmektedir. TEDAM Chrome Extension dışındaki tüm bileşenler tarafından ortak olarak kullanılır.

TEDAMEngine. Bu bileşen temel olarak, işlere bağlı senaryoların istemcilere dağıtılmasını yönetir. Kullanıcının başlattığı işler için devreye girerek, yeni başlayan ve çalışmaya devam eden işlerin, istemciler arasında belirli önceliklere bağlı olarak çalıştırılmasını sağlar. İşlere bağlı her bir senaryo, çalışma anında sırası geldiğinde önce TEDAMCore kullanılarak komutlara dönüştürülür. Dönüştürülen bu komutlar istemcilere yollanır. TEDAMAgent'larla olan bu iletişim tek taraflı olmayıp, TEDAMAgent'lardan gelen senaryoların çalıştırılma durumları TEDAMEngine tarafından kayıt altına alınarak, kullanıcı bilgilendirme ve raporlama işleri yapılır. TEDAMEngine'in bir diğer görevi de tanımlanmış REST arayüzü sayesinde Jenkins, Hudson, gibi sürekli entegrasyon (Continuous Integration) araçlarıyla entegre olabilir, böylece test edilen ürüne bağlı yapılandırma sürecine (build) dahil olabilir.

TEDAMAgent. Bu bileşen tamamen kendisine verilen görevleri yapmakla yükümlüdür. TEDAMAgent; TEDAMEngine ile sürekli iletişim halinde olup, kendisine iletilen komutları TEDAMCore bileşenini de kullanarak çalıştırır. Çalışma esnasında oluşan olumlu olumsuz durumları TEDAMEngine'e bildirir.

TEDAMFace. Bu bileşen kullanıcının yoğun olarak kullandığı arayüz uygulamasıdır. Kullanıcılar; test senaryolarını bu arayüzden sisteme girerler, snapshot dosyalarını yükleyip, adımlarla ilişkilendirirler, Test adım tiplerini seçerek, parametreleri hazırlayabilirler. Hazırlanmış olan senaryoların iş haline getirilmesi de yine TEDAMFace arayüzü üzerinden yapılır. Hazırlanan işlerin konfigüre edilmesi, çalıştırılması ve takibinin de yapıldığı bileşendir. Ayrıca kullanıcı yönetimi, proje yönetimi, TEDAMAgent'a gönderilen komutların yönetimi, komutlar için kullanılacak parametrelerin yönetimi, istemcilerin ve ortamların yönetimi gibi işler de TEDAMFace üzerinden yapılmaktadır.

TEDAM Chrome Extension. Kullanıcıların snapshot alabilmeleri için tasarlanmış bileşendir. Kullanıcı test edeceği uygulamanın, kullanıcıya yansıyan kısmındaki kodlamaya bağlı olarak kurallar tanımlar. Kurallar oluşturulduktan sonra, kullanıcı istediği herhangi bir ekran açıkken snapshot alabilir.

6 Sonuç

TEDAM'ın geliştirilip kullanılması ile beraber, otomasyon uygulanan yazılımdaki kod kapsamı kısa sürede, belirlenen ilk hedef olan %50 başarılmıştır. Test senaryo adımlarının atomik ve detaylı olarak yazılması, yapılan testin ne olduğunu anlaşılır kılmıştır. Test uzmanının otomasyon için herhangi bir kod yazma gereksiniminin olmaması daha az yetkin test uzmanlarının çalışabilir olmasını mümkün kılmıştır. Bu nedenle az kaynak ve zaman ile daha çok iş yapılması sağlanmıştır. Testlerin dağıtık olarak çalışmasına imkân sağlayan yapısı ile yüksek adetli test senaryolarının toplu olarak kısa sürede koşulabilmesi sağlanmıştır. Daha önceden tek bir senaryonun otomatikleştirilmesi günlerce sürebiliyorken, TEDAM ile birlikte bir günde yazılan senaryo sayısı yaklaşık 4 adet olabilmektedir. Bu kazanılan hız sayesinde otomatikleştirilen senaryo adedi 5000 adetlere gelmiştir.

Kaynaklar

1. Schmidt F. Richard: "Software Engineering: Architecture-driven Software Development",72
2. TURKEY SOFTWARE QUALITY REPORT 2018-19
3. Why You Shouldn't Skip Regression Testing, https://testlio.com/blog/shouldnt-skip-regression-testing/
4. Test Automation for Web Applications, https://www.seleniumhq.org/docs/01_introducing_selenium.jsp#to-automate-or-not-to-

automate
5. Why Automated Testing?, https://smartbear.com/learn/automated-testing/
6. Deliver faster and better software using test automation, https://www.atlassian.com/blog/add-ons/deliver-faster-and-better-software-using-test-automation
7. Top Challenges in Test Automation http://toolsqa.com/blogs/test-automation-challenges/
8. Data Driven Framework http://toolsqa.com/selenium-webdriver/data-driven-testing-excel-poi/