Test Otomasyonunda Karşılaşılan Zorluklar ve Öğrenilen Dersler: Telekomünikasyon Sektöründen Deneyimler

Gökhan Büyükyumukoğlu[1], Ersin Ersoy[1], Mehmet Şevket Özdemir[1], Selami Bağrıyanık[1], Adem Karahoca[2]

[1] Turkcell, İstanbul, Türkiye
{gokhan.buyukyumukoglu, ersin.ersoy, ozdemir.mehmet, selami.bagriyanik}@turkcell.com.tr
[2] Bahçeşehir Üniversitesi, İstanbul, Türkiye
adem.karahoca@eng.bau.edu.tr

Abstract. Yüksek kalitede yazılım üretmek gerçekleştirilmesi zor bir hedeftir. Yazılım kalitesi kaynaklı sorunlar, yazılım endüstrisi için önemi her geçen gün artmaya devam eden, çözülmesi zor, karmaşık ve çok boyutlu bir problem alanı olmaya devam etmektedir. Yazılım tabanlı ürün ve servislerdeki hatalar firmaların itibarları, finansal performansları ve müşterilerinin bağlılığı üzerinde doğrudan etkiye sahiptir. Gereksinim analizi, tasarım veya gerçeklenmiş yazılım kaynaklı hatalar, müşteri memnuniyetini olumsuz şekilde etkilemekte, firmanın uymakla yükümlü olduğu regülatif kurallara uyamaması ve ceza riski ile karşı karşıya kalmasına sebep olabilmektedir. Bazı yazılım hataları operasyonel performans, ölçeklenebilirlik, kullanırlık ve güvenlik sorunlarından da sorumludur. Düşük kalite aynı zamanda çalışan motivasyonu, ekip üretkenliği ve pazara çıkış hızı üstünde de çok önemli olumsuz etkiler yaratmaktadır. Test otomasyonu, bahsi geçen sorunun adreslenmesinde kullanılan çözümlerden biridir. Bu çalışmada, bir teknoloji ve iletişim hizmetleri sağlayıcı firmanın bir portföyünün web tabanlı uygulamaları kullanılarak yapılan servis, tarife ve kampanya testlerinin test otomasyonu dönüşümünde yaşanan deneyimler aktarılmıştır. Söz konusu deneyimler dört temel boyutta ele alınmıştır. Bu boyutlardan birincisi kullanılan teknoloji ve tasarım, ikincisi yazılım yaşam döngüsünü oluşturan süreçlere entegrasyon, üçüncüsü hem otomasyonu yapılan uygulamaların geliştirmesinde hem de otomasyon yazılım ve testlerinde çalışan test geliştiricilerinin ve uygulama kullanıcılarının motivasyonu ve sonuncusu da değişiklik yönetimi boyutlarıdır.

Keywords: Test Otomasyonu, Süreç Otomasyonu, Endüstriyel Deneyimler.

Challenges and Lessons Learned in Test Automation: Experiences from Telecommunications Industry

Gökhan Büyükyumukoğlu[1], Ersin Ersoy[1], Mehmet Şevket Özdemir[1], Selami Bağrıyanık[1], Adem Karahoca[2]

[1] Turkcell, İstanbul, Türkiye
{gokhan.buyukyumukoglu, ersin.ersoy, ozdemir.mehmet, selami.bagriyanik}@turkcell.com.tr
[2] Bahçeşehir Üniversitesi, İstanbul, Türkiye
adem.karahoca@eng.bau.edu.tr

Abstract. Producing high quality software is a hard-to-achieve target. Problems arising from software quality continue to be a difficult, complex, and multidimensional problem area that continues to grow in importance for the software industry. Errors in software-based products and services have a direct impact on the company's reputation, financial performance and customer loyalty. Requirement analysis, design or software based errors negatively affect customer satisfaction and can cause the firm to fail complying with regulations and face criminal fines. Some software faults are also responsible for operational performance, scalability, usability, and security issues. Low quality software also creates very significant negative effects on employee motivation, team productivity and time to market speeds. Test automation is one of the solutions used to address such problems. In this study, experiences from the test automation of service, tariff and campaign tests of a web-based application of a portfolio of technology and communication services provider firm is presented. These experiences are discussed in four basic dimensions. The first of these dimensions is used technology and design, the second is the integration to the processes that make up the software life cycle, the third is motivation of both test automation developers and application users working in automation software and tests and the last dimension is change management.

Keywords: Test Automation, Process Automation, Industry Experiences.

1 Giriş

Tüm yazılım projelerinde üretilen yazılımların testi vazgeçilmez bir gerekliliktir. Birim testleri, performans testleri, entegrasyon testleri, API testleri ve erişilebilirlik testleri bunlardan bazılarıdır.
Bu çalışmada uygulama ekranlarının test edilmesini kapsayan fonksiyonel testler üzerinde durulmuştur. El ile yapılan testlerde bir test uzmanı uygulamayı çalıştırır ve uygulama ekranlarında çeşitli senaryoları dener, değişik girdiler kullanarak uygulamanın beklenen davranışları sergileyip sergilemediğini, iş ve teknik gereksinimleri karşılayıp karşılamadığını test eder. Günümüzde yazılım endüstrisindeki test mühendisliği farkındalığı artırıp iyi test pratiklerinin uygulanmasını sağlasa da her projede belli seviyelerde yazılım hataları ortaya çıkmaktadır. Yazılım projelerinin kodlama aşamasında çıkan hata sayılarına bakıldığında Amerika Birleşik Devletleri ortalaması işlev puanı (function point) başına 1.75 yazılım hatasıdır [15].

Test otomasyonu ise kullanıcının yaptığı testlerin otomasyon yazılımları aracılığıyla bilgisayar tarafından yapılmasıdır. Test otomasyonu ile yazılım test hızı, test kapsamı, test maliyeti, yazılım kalitesi ve pazara sürüm süresi kriterlerinde kazanımlar sağlanmaktadır. Genellikle test otomasyonu çok sık tekrarlanan, yapılması zor olan ve kritik olan testler için geliştirildiğinden dolayı elle test pratiklerini tamamen ortadan kaldırmayı hedeflememektedir. Çok nadir yapılan testler için veya çok sık değişiklik gösteren sistemlerin testinde elle test yöntemleri kullanılmaktadır. Test otomasyon projeleri tıpkı diğer yazılım projeleri gibi devamlı bakım gerektirir, test edilen sistemler güncellendikçe, ilgili test otomasyon kodlarının da güncellenmesi gerekmektedir. Bu anlamda düşünüldüğünde test otomasyon projelerinde kapsamın belirlenmesi en önemli aşamalardan biridir. Günümüzde birçok test otomasyon yazılımları bulunmaktadır. Selenium [17], Sahi [18], HP UFT [19] bunlardan bazılarıdır. Test otomasyon projelerinde bu araçların desteklediği özellikler dikkatle incelenmeli ve test edilecek sistemler ile uyumlu olup olmaması dikkate alınmalıdır. Kurulum gerektirip gerektirmediği, dokümantasyon yeterliliği, otomasyon kodlarının test edilecek sistem kullanılırken kayıt yapılarak otomatik olarak oluşturulabilmesi gibi özellikler endüstride bu tip araçları değerlendirirken bakılan özelliklerden bazılarıdır [16]. Test otomasyon ürünlerinde bulunan özelliklerden bir tanesi de teknik olmayan kişilerin de test otomasyon yapabilmelerini sağlayacak yapılar içermeleridir. Buna rağmen birçok projede yazılım geliştirici seviyesinde kodlama bilgi ve becerisine sahip test uzmanlarına ihtiyaç duyulmaktadır. Hem bu yazılım araçlarının ücretli olabilmesi, hem de geliştirme ve bakım aşamasında teknik personele ihtiyaç duyulması göz önüne alındığında, test otomasyon projelerinin yatırım ve kaynak gerektirdiği görülmektedir.

Çalışmada, kazanılan deneyimler teknoloji, yazılım yaşam döngüsü, motivasyon ve değişiklik yönetimi başlıkları altında gruplanmıştır. Makalenin ikinci bölümünde test otomasyonu ile ilgili literatür verilmiştir. Üçüncü bölümde firmadaki projenin detayları tanıtılmıştır. Dördüncü bölümde karşılaşılan zorluklar ve öğrenilen dersler aktarılmış, son bölümde ise sonuçlar ve gelecek çalışmalara ilişkin planlar paylaşılmıştır.

2 İlgili Çalışmalar

Test otomasyonu, 25 seneyi aşkın bir geçmişe sahiptir ve test koşturma alanında belli bir olgunluğa erişmiştir [1]. Test otomasyonu maliyetli bir etkinlik olmasına rağmen doğru kapsam, süreç ve araçlarla tasarlandığında yatırımın geri dönüşünün alınabileceği uzun bir süredir bilinmektedir [6,7]. Hız, maliyet ve kaliteye olumlu etkileri de bazı yeni çalışmalardaki bulgularca da desteklenmiştir [5,11]. Ayrıca, test otomasyonuna konu olan senaryoların sistem modelinden yarı otomatik bir yöntemle türetilmesi ve türetilen senaryoların kodlamaya gerek olmaksızın otomatik çalıştırılmasını sağlamak gibi daha ileri araştırmalar gerçekleştirilmiş ve test zamanında kısalmalar tespit edilmiştir [14]. Bununla birlikte yazılım endüstrisinde otomasyon kullanımı buna koşut olarak yaygınlaşamamıştır. Örneğin geniş bir mobil uygulama kümesi üzerinde yapılan bir araştırmada, test otomasyonu pratiklerinin kullanılmadığı, mobil uygulamaların sadece % 9'unun koşulabilir test senaryolarına sahip olduğu ve test kapsamının % 40 civarında olduğu tespit edilmiştir [3]. Türkiye ve Norveç yazılım endüstrilerinde yapılmış iki araştırmanın çıktılarından biri, test otomasyonundan yeterince faydalanılmadığı sonucu olmuştur [4,13]. Bir başka çalışmada anahtar başarı faktörlerinin önemi konusunda akademisyenler ile profesyoneller arasında algı farklılıkları bulunduğu sonucuna varılmıştır. Sözü edilen çalışmada [8] akademik literatür daha çok planlama aktivitelerine en büyük önemi verirken, pratisyenler adanmış ve yetkin bir ekibin en önemli başarı faktörü olduğuna inanmaktadır. Demiryolu ve telekomünikasyon sektörlerinde gerçekleştirilen iki vaka çalışmasından yola çıkarak test otomasyonunda karşılaşılan güçlükleri detaylı bir şekilde irdeleyen ve çözüm önerileri sunan yakın tarihli bir çalışmada öne çıkan iki temel sorun ekip içindeki iletişim ve kaynak eksiklikleri olarak belirlenmiştir [9]. Günümüzde DevOps ve çevik yöntemlerin yaygınlaşmaya başlaması otomasyonun dinamiklerini değiştirmiş, baş edilmesi gereken yeni zorluklar çıkarmış ve önemini daha da arttırmıştır. Örneğin Scrum yöntemiyle çalışan ekipler için otomasyon seçime bağlı olmaktan çıkmış ve neredeyse zorunlu hale gelmiştir [2]. Test otomasyonunun sürekli teslimat (Continuous Delivery) içindeki kullanımına dair Avusturyalı bir firmanın deneyimlerini aktaran bir çalışmada yaklaşık 6 sene süren sürekli teslimat hattı oluşturma sürecinde test otomasyonun önemi vurgulanmaktadır [10]. Test otomasyonu ve DevOps arasında simbiyotik bir ilişki olduğu söylenebilir. Finlandiya'daki üç yazılım geliştime organizasyonunda gerçekleştirilen bir vaka çalışmasında DevOps pratiklerinin de test otomasyonunu geliştirdiği belirlenmiştir [12]. Aynı çalışmada, sürekli teslimat hattı kurarken ve işletirken dikkat edilmesi gereken anahtar başarı faktörlerine de yer verilmiştir. Bu faktörlerin başında yönetim sponsorluğunun alınması, sürekli teslimat hattı için kollektif sorumluluk alınması, test ortam yönetiminin sağlanması, yazılımın test edilebilirliğinin arttırılması ve test veri yönetiminin oluşturulması gelmektedir [10].

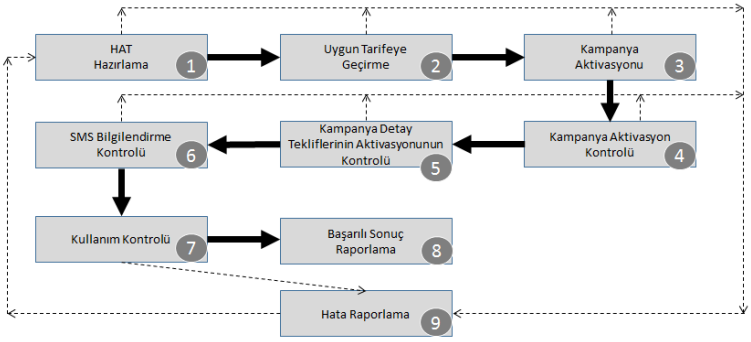
Bu çalışmada ise test otomasyonunun yazılım geliştirme döngüsündeki klasik konumlandırmasından farklı bir şekilde kullanılması ele alınarak literatüre katkı sağlanmıştır. Sözkonusu senaryoda test otomasyonu, bir çok farklı uygulamanın bütünleşik olarak çalıştığı bir telekomünikasyon iş destek altyapısı üzerinden yapılan karmaşık servis, tarife ve kampanya tanımlarının canlı ortam testinde kullanılmıştır.

Yani buradaki temel motivasyon tek bir ürün veya uygulamanın geliştirilmesi esnasında test otomasyonu kullanmak değil, eş zamanlı olarak geliştirilen bir çok entegre uygulamanın canlıya alınması ile bir araya gelmesinden oluşan karmaşık sürecin içindeki etkileşimlerin sağlığının sürekli kontrol edilmesidir. Böylece hem elle yapılan işin gerektirdiği efordan kaynaklı verimlilik kayıpları hem de insan hatalarının yol açtığı risklerin azaltılması amaçlanmaktadır.

3 Endüstriyel Vaka Çalışması

Bu bölümde firmada gerçekleştirilen test otomasyon projesinin kapsamı açıklanarak bir örnek senaryo ile detaylandırılmıştır. Çalışma, bir teknoloji şirketinin müşterilerine sunacağı servis, tarife ve kampanyalar ile ilgili tanımlamaların son kullanıcılara sunum öncesi yapılan doğruluk testlerinin gerçekleştirildiği süreçlerin otomasyonunu amaçlamaktadır. Projede SahiPro ve Java programlama dili kullanılmıştır.

Yazılım mühendisliğinde test otomasyonu denildiğinde ilk akla gelen, belirli bir uygulamanın uçtan uca otomasyonudur. Bu çalışmada ise belirli bir uygulamanın değil belirli bir servis, tarife ya da kampanyanın doğruluğunu teyit etmek için hali hazırda uygulanan süreç kapsamında kullanılan tüm uygulamaların otomasyonu kapsanmaktadır. Şekil 1 de örnek bir kampanya için hazırlanmış test otomasyon senaryosu akışı gösterilmiştir. Söz konusu test senaryosu 5 farklı uygulama, 12 farklı ekran üzerinde çalışmaktadır.



Şekil 1. Örnek bir test otomasyon akışı

Şekil 1. deki akışta, öncelikle test sırasında kullanılacak hat uygun duruma getirilmektedir. Bu adımda hattın daha önce çalıştırılan test senaryolarından kaynaklı olarak üzerine tanımlanmış özelliklerinin ilk günkü haline getirilmesi sağlanmaktadır (1). Sonrasında söz konusu kampanya aktivasyonu için ön şart olan hattın A tarifesinde olma koşulunu sağlamak için ilgili test hattı A tarifesine getirilmektedir. Bu getirme işlemi birden fazla ekran üzerinde, birden fazla işlemin gerçekleştirilmesi ile tamamlanır (2). Bir sonraki adımda söz konusu kampanyanın aktivasyonu kampanya aktivasyon ekranından tanımlanır (3). Kampanya aktivasyonunun başarıyla

tamamlandığı, aktivasyon kontrolü ekranından kontrol edilir (4). Kampanya aktivasyonları her durumda gerçek zamanlı olarak yapılmadığından, ekran üzerinde belirli bir süre bekleme gibi akışlar uygulanır. Sonrasında kampanya aktivasyonu kapsamında hat üzerine tanımlanması gereken detay özelliklerin (örneğin 100 dakika, 100 kısa mesaj gibi) verildiği kontrol edilir (5). İlgili kampanyanın aktiflendiğine dair bilgilendirme kısa mesajı yollandığı kontrol edilir (6). Belirli senaryolarda hat üzerinden kullanım yapıldığında aktiflenen kampanyanın kullanıldığının da kontrol edilmesi gerekebilmektedir. Böyle durumlarda son olarak kullanım kontrolleri yapılır (7). Başarılı bir senaryo sonrası sonuçların raporlanması gerçekleştirilir (8). İnce çizgiler ile gösterilen akışta, herhangi bir adımda hata alınması durumunda hata raporlanır ve senaryo tekrar denenmek üzere ilk adıma gider (9).

4 Güçlükler ve Öğrenilen Dersler

Bu bölümde test otomsayon projesinin ilk fazında elde edilen deneyimler, teknoloji ve tasarım, motivasyon, değişiklik yönetimi ve yazılım yaşam döngüsü başlıkları altında açıklanacaktır.

4.1 Teknoloji ve Tasarım

Test Otomasyon Senaryolarının Arka Planda Çağırılan Servisler ile Etkileşimi. Senaryoların el ile test edilmesi sırasında web tabanlı uygulamaların kullanıcı arayüzlerine bir internet tarayıcısı vasıtası ile girilir ve ekranlarda işlemler yaparak bu testler gerçeklenir. Bu işlemler sırasında kullanıcı, sisteme ve verilere hakim olduğu için bazı kararlar verir. Örneğin bir kampanyanın sadece faturalı hatlara tanımlı olup olmaması bilgisini bilir ve ona göre ekranı kullanır. Ancak test otomasyon yapılırken sistem bu bilgiyi bilmemektedir. Bu durumlarda arka planda bir başka sisteme bir arka plan çağrısı yapılarak bu bilgi edinilir ve dönen değere göre de otomasyon senaryosu ekrandaki akışa karar verir. İşte bu çağırılan servislerin sayısı çoğaldıkça test otomasyon sisteminin diğer sistemlere olan bağımlılıkları artar. Bu bağımlı olunan servislerdeki her aksama, test otomasyon senaryolarının aksamasına yol açar ve sistemi kırılgan ve güvenilmez bir hale getirebilir. Bu arka plan servislerinin doğru çalışması ve sayısı çok önemlidir. Kırılgan ve ayakta olma zamanı düşük olan servisler test otomasyon senaryolarına dahil edilmemelidir. Dahil edilmesi zorunlu olan durumlarda da ilgili servisin sahibi ile servislerin sürekli hazır olması konusunda mutabakat sağlanmalıdır.

Test Otomasyon Ekran Görüntüsü Alınırken Karşılaşılan Siyah Ekran Sorunu. Test otomasyon senaryolarının kullanıcıların bilgisayarından değil uzaktan bağlantı ile Windows tabanlı sunucularda koşması durumunda eğer sunucuya aktif olarak bir bağlantı bulunmuyorsa, senaryo kapsamında yapılan işlemlerin ekran görüntüsü alınması sırasında sunucunun aktif bir görüntülü desktop ortamı olmadığı için siyah ekran görüntüsü alınmaktadır. Burada sunucuya her zaman bağlı bir istemci

bulundurmak veya sunucu üzerine özel SanalAğ İşleme (VNC: Virtual Network Computing) sunucusu yazılımları kurmak gibi alternatif çözümler kullanılmalıdır.

Senaryolarda HTML Komponentlere ID Alanı Dışında Erişim. Web tabanlı uygulamarın test otomasyonunun yazılması sırasında ekrandaki belirli komponentlere tıklamak, üzerine gelmek veya bir şekilde bu komponentleri kullanmak gerekmektedir. Bunu yapabilmek için test otomasyonu kodlarının bu komponentlere erişmesi gerekmektedir. Burada doğru yöntem bu komponentlere doğrudan ID alanları üzerinden erişmektir. ID tekil bir alandır ve ilgili komponente erişimde garantili bir yöntemdir. Erişilmek istenen HTML komponentlere ait ID alanı uygulama kodlarında tanımlanmamışsa ilk önce otomasyonu yazılacak uygulamanın kodlarının değiştirilmesi ve erişilecek komponentlere ID tanımlatılması sağlanmalıdır. Komponentlere, ID alanı dışında etiket veya ekrandaki yakınında bulunan diğer komponentlerin pozisyon bilgisinden erişilebilir ancak bunlar doğru erişimler değildir. Çünkü bu bilgilerin hepsi değişime açıktır. İlgili komponentin ekrandaki yeri değişebilir, veya üzerindeki etiket değiştirilebilir.

Senaryoların Yazımında Kullanılan Teknolojinin Tarayıcı Değiştirme Özelliği. Otomasyon kapsamında kullanılan uygulamaların sadece belirli tarayıcılar üzerinde çalışması gerekebilmektedir. Örneğin A uygulaması sadece Chrome tarayıcısında çalışabilirken B uygulaması ise Firefox tarayıcısını gerektirebilir. Bu durumda test otomasyon geliştirmelerinin yapıldığı teknolojinin senaryonun ortasında kullanılan tarayıcıyı değiştirebilme özelliği olması önem kazanmaktadır. Test otomasyon projelerine başlamadan bu konuya ayrıca dikkat edilmelidir. Bu özelliğin olmaması durumunda farklı tarayıcı gerektiren uygulamaların olduğu senaryolar ikiye bölünerek çözülmüştür.

Test Otomasyon Senaryolarında Adımlar Arası Sabit Beklemeler. Test otomasyon senaryo yazımlarında bazen bir işlemden sonra ekranda bazı değişikliklerin olması beklenir ki ikinci bir adıma geçilebilsin. Bir kaydetme işleminden sonra bir veri tabanı tablosu üzerinde oluşan kaydın işlenmesinin beklenmesi bu duruma örnek olarak verilebilir. Bu gibi durumlarda adımların arasına sabit t saniye bekleme gibi komutların eklenmesi durumunda otomasyon senaryolarının süresi uzamaktadır. Senaryolarda sabit bekleme komutları yerine, işlemin tamamlandığının anlaşılıp senaryoya devam edilmesinin sağlanması senaryo süresine katkı sağlayacaktır. Kullanılan teknolojilerin wait() metodları değil mümkün olduğu her durumda waitFor() veya wait($condition) metodları kullanılmalıdır. SahiPro teknolojisi düşünüldüğü zaman wait(5000) komutu yerine Onayla butonu görünür olana kadar bekle anlamına gelen wait(5000, _isVisible(_button("Onayla"))) komutu tercih edilebilir.

4.2 Yazılım Yaşam Döngüsü ve Otomasyonun Entegrasyonu

Senaryo Analizlerinin Eksik İletilmesi. Otomasyon kapsamında kullanılan uygulamalarda birçok iş kuralı bulunmaktadır. Bu iş kurallarının bazıları ise çok nadir olarak oluşan durumlarda ortaya çıkmaktadır. (Örneğin sadece son dört aydır faturası ödenmemiş bir hat için ekrana bir pencere çıkması gibi). Bu pencere çıkması gibi nadir durumlar senaryo tariflenirken unutulmakta ve senaryo kodları içine doğal olarak programcı tarafından eklenmemektedir. Bu durumda canlı ortamda koşulan test otomasyon senaryolarında bu uç örnekler yaşandığı zaman senaryo başarısız olmakta ve tekrar kod yazılması gerekmektedir. Bu örneklerin sayısı çoğaldıkça test otomasyon senaryolarının güvenilirliği azalmakta, sürekli hata alan bir sisteme dönüşmektedir. Bu durumda da sisteme güven azalmaktadır.

Senaryolar için Test Kapsama Oranının Düşük Olması. Her geliştirme sürecinde olduğu gibi test otomasyon senaryolarının da geliştirme aşaması bittikten sonra test edilmesi gerekmektedir. Bu testlerde genelde olağan akışın test edilmesi durumlarında uç örnekler gözden kaçabilmektedir. Test otomasyon senaryolarının hata aldığı durumlar genelde canlı ortamda çok nadir rastlanan iş akışlarıdır. Bunların test aşamasında bulunması için ise ilgili verinin hazırlanması ve ilgili iş akışının detayının test ekibi tarafından da bilinmesi gerekmektedir. Bu analiz aşaması detay bilgisinin test ekibi tarafından bilinmediği durumlarda bu nadir görülen iş akışı hataları hep canlı ortamda çıkmakta ve sisteme olan güven azalmaktadır.

Test Verisi İhtiyacı. Test otomasyon senaryosu kapsamı büyüdükçe senaryoyu çalıştırabilmek için gerekli test verilerinin hazırlanması da zorlaşmaktadır. Test otomasyon senaryoları geliştirilirken aynı zamanda test otomasyon senaryolarının kendi test verilerini oluşturması, gerek motivasyon gerekse de elle hazırlanan test verilerindeki eksiklikler sebebi ile oluşacak hataların önüne geçebilir.

Test Otomasyonu Yazılacak Uygulamaların Ortam Farklılıkları. Test otomasyon senaryoları yazımı aşamasında test geliştiricileri bu uygulamaların canlı ortamına erişememektedir, bu sebeple genelde test ortamlarını kullanmaktadırlar. Ancak bazı uygulamalarda test ortamlarında çalışan kod ve ekranlar canlı ortamla farklılık gösterebilmektedir. Bu durumda da yazılan senaryolar canlı ortamda çalıştığında hatalar ile karşılaşılabilmektedir. Bir uygulamanın test otomasyonu yazılmaya başlanmadan önce ortam sorunlarının giderilmesi gerekmektedir.

4.3 Test Geliştiricisi ve Kullanıcı Motivasyonu

Test Otomasyon Senaryo Kapsamı. Test otomasyonu yapmaktaki amaçlardan bir tanesi de insan kaynağı olmadan ilgili yazılımın testinin bilgisayar sistemleri tarafından belirlenen periyotlarda otomatik olarak yapılmasını sağlamak ve otomasyon sonucuna göre aksiyon almaktır. Burada başarılı olabilmek ve insan faktörünü ortadan kaldırabilmek için test otomasyonu senaryolarının test kapsamının büyük çoğunluğunu kapsaması gerekmektedir. Vakamızda karşılaşılan en büyük

sorunlardan bir tanesi proje başlangıcında belirlenen senaryoların tüm servis, tarife ve kampanyaların sadece belirli bir miktarını karşılamasıdır. Bu durumda bir servisimizi kullancılarımıza sunmadan önce koşulan test otomasyonu tek başına bir güvence sunamamakta, bu nedenle el ile otomasyon senaryolarının kapsamadığı kısımların test edilmesi gerekmektedir. Burada iki farklı test sistematiği kullanımının test yapan çalışanlar için bir zorluk oluşturduğu görülmüştür. Sadece test otomasyonuna güvenilmemesi sürekli olarak tüm testlerin aynı zamanda el ile de yapılmasını gerektirmektedir. Bu durumda da test otomasyon sistemi ve ürettiği raporlar her zaman çalışanlar için kontrol etmeleri gereken ikinci bir iş olarak kalabilmektedir. İnsan faktörü olmadan test otomasyonu ile de canlı ortama rahatlıkla servis çıkılması hedefi gerçeklenememektedir. Test otomasyonu kodlarının en azından bir sistemin belirli kısımlarını güvenli bir şekilde test ettiği, basit ve tekrarlanan testlerin tamamını kapsadığı bir ortam sağlanabilirse, kullanıcıların bu kısımları hiç elle test etmemeleri ve sadece çok zor ve karmaşık senaryolara odaklanmaları kullanıcı motivasyonunu arttırabilmektedir. Test otomasyonunun aynı zamanda sistemlerin temel ve hiç değişmeyen kısımlarının testi işini başarıyla yaptığı durumlarda, yazılım test uzmanları el ile sadece uygulamalara yeni eklenen özellikleri ve en son geliştirmeleri test ederler, bu da testlerde sıkıcılığın önüne geçerek motivasyonu arttıran bir faktör olarak öne çıkmaktadır.

Test Otomasyon Senaryolarının Çalışması Sırasında Hata Alması. Elle test sırasında akışın herhangi bir tanesinde bir sorunla karşılaşıldığında test eden kişi bu sorunu aşmak için uğraşmakta, başka kişilere sorular sorabilmekte ve yardımlarını alabilmektedir. Sorun çözüldüğü anda işlemine kaldığı yerden devam etmekte ve o ana kadar yaptığı işlemler ve tanımladığı veriler boşa gitmemektedir. Uçtan uca bir test otomasyonunda ise bu tip beklenmedik bir durum ile karşılaşıldığında süreç tıkanmakta ve senaryo başarısız olarak sonlanmaktadır. Daha sonra başarısızlık sebebi otomasyon logları incelenerek test geliştirici kişiler tarafından düzeltilmekte ve senaryo tekrardan koşulmaktadır. Burada senaryo en baştan tekrar başladığı için test otomasyon ile test edilen süreç elle teste göre daha uzun sürmekte ve test uzmanlarının aynı iş üzerindeki çalışma saatlerini arttırarak kişinin motivasyonunu düşürebilmektedir. Bu noktada uçtan uca bir çözüm yerine hata alınana kadar devam eden bir yapı ve hata alındığı noktada elle teste devam edilmesini sağlayan bir sistem geliştirilmesi bu sorunun çözümüne yardımcı olacaktır.

4.4 Değişiklik Yönetimi

Test Otomasyon Uygulama Ekranlarının Sık Değişmesi. Test otomasyon senaryolarının yazılması tıpkı yazılım geliştirme aşamasında kod yazmaya benzemekte ve benzer zorluklar taşımaktadır. Bu durumda da bir senaryonun yazılması zaman alabilmektedir. Uygulama ekranlarının çok sık değiştiği ve oturmuş ekranlar olmadığı durumlarda ise ekrandaki değişiklikleri hemen aynı anda test otomasyon senaryolarına da yansıtmak gerekmekte aksi takdirde canlı ortamdaki ekranın versiyonu ile otomasyon senaryosu uyuşmamakta ve hatalı test sonuçları üretilmektedir. Ek olarak ekrandaki değişiklikleri test otomasyon kodlarına

yansıtmanın zorluğu ve zaman alması dışında karşılaştığımız bir başka zorluk ise ekranların değiştiği bilgisine erişmekte yaşanmıştır. Şekil 1. deki akış üzerinde anlatıldığı gibi belirli bir servis, tarife veya kampanyanın test edilebilmesi için şirketteki farklı birimler tarafından geliştirilen bir çok uygulamanın kullanılması gerekmektedir. Otomasyon senaryoları ise bu birimlerin dışında ayrı bir birim tarafından yazılmıştır. Burada birimler arası iletişimin önemi öne çıkmaktadır. İletişimin yanısıra versiyon kontrol sistemlerinin düzenli olarak kontrol edilmesi ile değişen ekranların önceden belirlenmesi ve test otomasyon ekibine otomatik olarak iletilmesi gibi ihtiyaçlar olduğu görülmüştür.

5 Sonuçlar ve Gelecek Çalışmaları

Çalışmamızın ilk fazı sonucunda bir çok farklı nedenden dolayı çeşitli zorluklar yaşanmış ve bu zorlukların neticesinde kazanımlar elde edilmiştir. Bu kapsamda teknoloji ve tasarım konusunda daha basit bir mimari ve teknoloji kullanımının çalışmanın başarısını arttıracağı görülmüştür. Bu kararın sonucunda kullanıcılara daha fazla sorumluluk verilmesi kaçınılmazdır ancak gerçekleştirdiğimiz vaka çalışmamızda tamamiyle otomasyon yerine belirli noktalarda kullanıcılara inisiyatif vererek süreç otomasyonunun geliştirilmesinin başarı oranını arttıracağı görülmüştür. Özellikle senaryonun hata alması durumunda kullanıcının işlemi el ile gerçekleştirip kaldığı yerden senaryonun devam etmesini sağlamasının çok önemli bir kazanım olacağı görülmüştür. Buna ek olarak test otomasyon senaryosunun çok iyi analiz edilmesi gerektiği, sadece ana akışların değil detay akışların da dikkate alınması gerektiği ortaya çıkmıştır. Benzer şekilde değişiklik yönetiminin iyi yapılamamasının senaryoların hata almasında bir etken olduğu, bu nedenle kullanılan uygulama ekranların değişiminin proaktif bir şekilde belirlenip önlemler alınması gerektiği ortaya çıkmıştır. Bütün bu sebeplerden dolayı test otomasyon senaryolarının koşulması sırasında yaşanan hataların gerek test geliştiricisi gerekse de kullanıcı motivasyonunu olumsuz etkilediği de gözlenmiştir.

Bu çalışma kapsamında kazanılan deneyimler çalışmanın bundan sonraki fazlarında kullanılacaktır. Öncelikli iki senaryo için öğrenilen derslerden yola çıkarak gerekli iyileştirmeler yapılacak, iyileştirmelerin başarılı olması durumunda diğer senaryolara da uygulanacaktır.

Kaynaklar

1. Garousi, V., Elberzhager, F. : “Test Automation : Not Just for Test Execution”. IEEE Software, 34(2), 90-96. (2017)
2. Tyagi, S., Sibal, R., Suri, B. : “Adopting Test Automation on Agile Development Projects: A Grounded Theory Study of Indian Software Organizations”. In : International Conference on Agile Software Development (pp. 184-198). Springer, Cham. (2017)
3. Kochhar, P. S., Thung, F., Nagappan, N., Zimmermann, T., Lo, D. : “Understanding the test automation culture of app developers”. In : Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on (pp. 1-10). IEEE. (2015)

4. Garousi, V., Coşkunçay, A., Betin-Can, A., Demirörs, O. : “A survey of software engineering practices in Turkey”. Journal of Systems and Software, 108, 148-177. (2015)
5. Kumar, D., Mishra, K. K. : “The Impacts of Test Automation on Software's Cost, Quality and Time to Market”. Procedia Computer Science, 79, 8-15. (2016)
6. Hoffman, D. : “Cost benefits analysis of test automation”. STAR West, 99. (1999)
7. Ramler, R., Wolfmaier, K. : “Economics perspectives in test automation: balancing automated and manual testing with opportunity tutar”. In : Proceedings of the 2006 international workshop on Automation of software test (pp. 85-91). ACM. (2006)
8. Rodrigues, A., Dias-Neto, A. : “Relevance and Impact of Critical Factors of Success in Software Test Automation lifecycle: A Survey”. In : Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing (p. 6). ACM. (2016)
9. Tcholtchev, N., Schneider, M. A., Schieferdecker, I. : “Systematic Analysis of Practical Issues in Test Automation for Communication Based Systems”. In : Software Testing, Verification and Validation Workshops (ICSTW), 2016 IEEE Ninth International Conference on (pp. 250-256). IEEE. (2016)
10. Gmeiner, J., Ramler, R., Haslinger, J. : “Automated testing in the continuous delivery pipeline: A case study of an online company”. In : Software Testing, Verification and Validation Workshops (ICSTW), 2015 IEEE Eighth International Conference on (pp. 1-6). IEEE. (2015)
11. Alegroth, E., Feldt, R., Olsson, H. H. : “Transitioning manual system test suites to automated testing: An industrial case study”. In : Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on (pp. 56-65). IEEE. (2013)
12. Riungu-Kalliosaari, L., Mäkinen, S., Lwakatare, L. E., Tiihonen, J., Männistö, T. : “DevOps Adoption Benefits and Challenges in Practice: A Case Study”. In : Product-Focused Software Process Improvement: 17th International Conference, PROFES 2016, Trondheim, Norway, November 22-24, 2016, Proceedings 17 (pp. 590-597). Springer International Publishing. (2016)
13. Anika, S. : “Investigation of the use of test automation in software quality assurance in Norwegian companies and organisations”. (Master's thesis, NTNU). (2016).
14. Özkan, U., Sözer, H. : “Web uygulamaları için model bazlı test süreci otomasyonu”. In : 9th Turkish National Software Engineering Symposium, UYMS 2015. CEUR. (2015)
15. Jones, C. : Software Defect Origins and Removal Methods. (2012)
16. Sheth, T., Singh, S.K. : "Software Test Automation - Approach on evaluating test automation tools", International Journal of Scientific and Research Publications, Volume 5, Issue 8. (2015)
17. Selanium Test Otomasyon, http://www.seleniumhq.org/. Erişim Haziran.
18. Sahi Test Otomasyon, http://sahipro.com/. Erişim Haziran.
19. UFT Test Otomasyon, https://software.microfocus.com/en-us/software/uft. Erişim Haziran.