Kullanıcı Arayüzü Yazılımı Test Otomasyonundan Beklentiler ve Riskler

Ömer BİLGİN[1], Mustafa KASAPOĞLU[1]

[1] Radar Elektronik Harp ve İstihbarat Sistemleri (REHİS) Sektör Başkanlığı, ASELSAN A.Ş. Ankara

{omerbilgin, mkasapoglu}@aselsan.com.tr

Özet. Günümüz yazılım dünyasında, kaliteli yazılım beklentisinin artmasının yanı sıra, yetkin personel, zaman ve maliyet kısıtları nedeniyle yazılım test otomasyonu da gittikçe önem kazanmaktadır. Özellikle kullanıcı arayüzü, yazılımların son kullanıcıyla etkileşimde olan kısım olduğu için yazılım testlerinin daha detaylı ve hızlı yapılması beklenmektedir. Artan kalite ihtiyacına cevap verebilmek için yazılım otomasyon faaliyetlerine başlanmadan önce beklentilerin, hangi yaklaşım ve araçların kullanılacağının belirlenmesi önemlidir. Bu makalede; bir kullanıcı arayüzü yazılımının test otomasyonuna karar verilmesi, otomasyonun geliştirilmesi ve süreçlere uyarlanması üzerine yapılan çalışmada elde edilen deneyimler aktarılmıştır.

Abstract. As a result of rising quality demand, lack of qualified staff, time and resource led software test automation more important in software development industry. Due to the fact that UI is the most interacting point with end users, especially UI tests need to be done faster and more detailed. It's important to specify expectations, approach to be followed and tools to be used before starting test automation process. In this article; experiences gained during test automation decision, test automation development and adapting to the processes are explained.

Anahtar Kelimeler: Yazılım Kalite, Yazılım Test Otomasyonu, Kullanıcı Arayüzü Testi, Yazılım Geliştirme Süreç İyileştirmesi

Keywords: Software Quality, Software Test Automation, GUI Test, Software Development Process Improvement

1 Giriş

Yazılım test otomasyonu bir zorunluluk değil ihtiyaçtır. Bu makalede test otomasyonuna duyulan ihtiyacın nasıl belirleneceği ve sonrasında yapılması gerekenler yaşanılan tecrübeler doğrultusunda anlatılmıştır.

2 Test Otomasyonuna Duyulan İhtiyacın Belirlenmesi ve Test Otomasyonundan Beklentiler

Yazılım test otomasyonu ile temelde tekrarlı iş kalemlerinin minimuma indirilmesi, daha kararlı ve tekrar edebilen test senaryolarının oluşturulması/koşturulması ve bu işlemler esnasında harcanan işçiliğin düşürülmesi beklenmektedir. Ayrıca manuel test ile yapılamayacak bazı test senaryoları test otomasyonu ile gerçekleştirilebilir. Örneğin kullanıcı arayüzünde bir insanın verebileceği tepkiden daha hızlı bir tepki gerek olduğu durumlarda bu ancak yardımcı bir yazılım veya test otomasyonu ile testleri koşturmak mümkün olabilir. Test otomasyonu aynı zamanda çok fazla test girdisi ile yapılan tekrarlı testler için sıklıkla kullanılır. Yazılım test otomasyon faaliyetleri uzun vadede zaman ve iş gücü kazancı sağlaması gerekirken otomasyon faaliyetlerine başlamadan önce yapılan yetersiz ölçüm ve ön çalışmalar nedeniyle zaman, iş gücü ve hatta ilgili ekibin motivasyon ve itibar kaybına neden olmaktadır. Yazılım test otomasyonu zaman, sabır ve bilinçli bir ekip isteyen zorlu bir süreçtir. Bu süreç içinde atılan adımların titizlikle seçilmesi başarıya giden yolda en büyük etkendir.

Yazılım test aracından doğru beklentiler ile sıkça düşülen hatalı beklentiler aşağıda belirtilmiştir.

Yazılım test otomasyonundan başlıca beklentiler:

- Regresyon testlerinin tekrar tekrar yapılabilmesi
- Testlerin daha sık koşturulabilmesi
- Manuel olarak yapılması zor ya da imkânsız testlerin yapılabilmesi
- Kaynakların daha iyi kullanılması
- Kararlı ve tekrar edilebilir testlerin gerçekleştirilmesi
- Testlerin tekrar kullanılabilmesi
- Sürümlerin daha erken yayımlanabilmesi
- Yazılım güvenilirliğinin artırılması

Yazılım test otomasyonundan beklenilmemesi gerekenler:

- Test aracının hatasız test yapmasını beklemek
- Test aracının manuel testleri iyileştirebileceğini düşünmek
- Yazılım test otomasyonunun tüm hataları bulacağını düşünmek
- Test aracının hatasız yazılım çıkarmasını beklemek
- Bir kez tanımlanan testlerin değişikliğe ihtiyaç duymayacağını düşünmek
- Test aracının organizasyona hızlı ve kusursuz uyum sağlayacağını düşünmek

3 Yazılım Test Otomasyonu için Proje Seçilmesi

Proje seçimi yapılırken seçilen yazılımın yazılım yaşam döngüsünün sonlarında olmamasına dikkat edilmelidir. Teslimatı yaklaşmış ya da teslim edilmiş, bakım

sürecine girmek üzere olan bir proje otomasyona geçirilecek proje olarak seçilmemelidir. Aynı şekilde henüz tasarımı olgunlaşmamış, yazılım yaşam döngüsünde başlarda olan bir projenin tasarımının süreç boyunca fazla değişiklik gösterebileceği düşünüldüğünde otomasyona geçirilecek proje olarak seçilmesi uygun değildir. Bu tip projeler yerine tasarımı olgunlaşmış, teslimat sürecine girmemiş bir projeyi otomasyona geçirilecek proje olarak seçmek en doğrusu olacaktır.

Ayrıca gereksinimleri ve senaryosal yapısı karmaşık olmayan bir yazılım ile pilot otomasyon çalışması yapılmalıdır. Böyle bir yazılımla çalışmalara başlamak ekibin yazılımın detaylarında boğulmadan doğrudan otomasyonu tecrübe etmeye yönelik çalışmasını sağlayacaktır. Bu durum yazılım test otomasyonunda görev alacak ekibin ayrıca bir eğitim alınmayacaksa test aracına ve sürece daha kolay adapte olmasını sağlar.

4 Yazılım Test Otomasyonu Maliyet ve Fayda Analizi

Yazılım test otomasyonu maliyetli bir süreçtir. Toplam maliyete kullanılacak test aracının lisans maliyeti, testlerin otomasyonu için harcanan işgücü(test aracının kullanımının öğrenilmesi ve testlerin otomasyona aktarılması için gereken işgücü) maliyeti ve otomasyona aktarılan testlerin bakımı için gereken işgücü maliyeti sayılabilir. Maliyet hesaplanmasında birçok bilimsel çalışma yapılmıştır. Aşağıda örneğini verdiğimiz hesaplama yöntemi yazılım test alanında yer etmiş "evrensel formül" olarak kabul edilir [5]. Bu formülün amacı manuel testlerin yapılması için gerekli maliyet ile otomasyona aktarılmış testlerin yapılması için gerekli maliyetin karşılaştırılmasıdır. Bu karşılaştırma sonucu test otomasyonu sürecinde en başta harcanan yüksek maliyetin kaç test tekrarı sonrasında karşılandığı ve her test tekrarında kar edilmesi tahmin edilen maliyet hesaplanır(Şekil 1).

Formülün detayları şöyledir:

V = Test tasarımı yapılması ve testlerin tanımlanması maliyeti

D = Bir testin koşturulması maliyeti

n = Test koşum sayısı

Buna göre, otomasyona geçirilen testin toplam maliyeti:

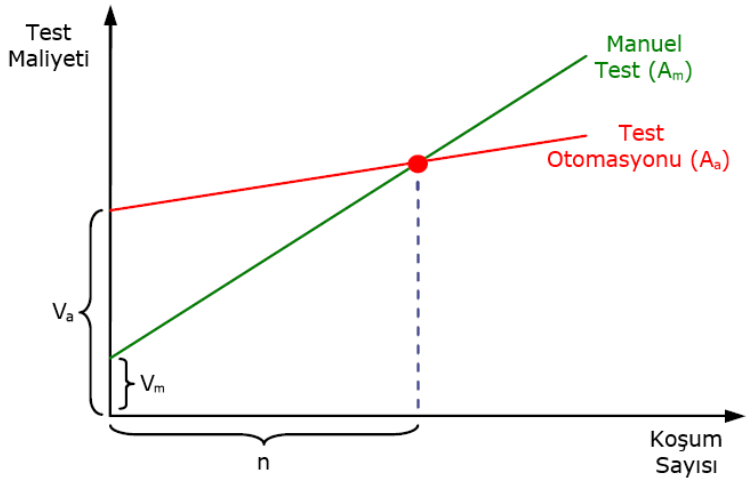
$A_a = V_a + n * D_a$

Burada V_a test otomasyonu tasarımı ve tanımlaması maliyeti, D_a tek bir otomasyon testi koşum maliyetidir.

Aynı şekilde manuel test toplam maliyeti:

$A_m = V_m + n * D_m$

Burada V_m manuel test tasarımı ve tanımlaması maliyeti, D_m tek bir manuel test koşum maliyetidir.



Şekil 1 Test Maliyeti Koşum Sayısı Grafiği

Bu iki formül kullanılarak başa baş noktası hesaplanır. Başa baş noktası testlerin otomasyona geçirilmesi kararının verilmesi esnasında maliyet açısından yapılması gereken en az koşum sayısını belirlemeyi sağlar. Böylece bulunan koşum sayısının altında kalınıyorsa ve maliyet dışında başka bir kıstas yoksa testlerin otomasyona geçirilmesi maliyet açısından mantıklı değildir.

Seçilmiş bir projede yapılan ölçümlerde; manuel test tasarımı yapılması ve testlerin tanımlanması 10 adam saat, her bir koşum maliyeti de 30 adam saattir. Otomasyon ile testlerin koşturulması için test tasarım ve tanımlama maliyeti 70 adam saat ve her bir koşum maliyeti de 10 adam saattir. Şekil 1'de gösterilen maliyet hesabı formülü kullanılarak yapılan hesaplamada Tablo 1 elde edilir.

Tablo 1 Maliyetler

Koşum Sayısı	Manuel Maliyet	Otomasyon Maliyet
1	40	80
2	70	90
3	100	100
10	310	170

Testler 3 koşumdan az yapıldığında testlerin maliyet açısından manuel yapılması anlamlı iken 3 koşum yapıldığında başa baş noktasına erişilir. 4 ve üzeri koşum sayısında test otomasyonunun daha az maliyetli olduğu görülmektedir. Koşum sayısı arttıkça test otomasyonunun maliyet açısından sağladığı faydanın daha fazla olduğu görülmektedir.

5 İhtiyacı Karşılayacak Test Aracının Araştırılması

Sektörde yazılım test otomasyon aracı olarak hem ücretli hem de ücretsiz oldukça fazla test aracı bulunmaktadır. Ücretsiz uygulamaların birçoğu web sitelerinin kullanıcı arayüzlerini test etmeyi hedeflemektedir. Bu aşamada dikkatle belirlenen ihtiyaçları karşılayacak en uygun araç seçilmelidir. Örneğin testlerini otomatize edeceğimiz yazılım Java ortamında geliştirilmişken Java grafik arayüz nesnelerini tanımayan bir araç seçilmemelidir. Organizasyon içi diğer geliştirme araçları ile uyumsuz bir aracı seçmektense uyumlu olan bir araç tercih edilmelidir. Benzer kıstaslar, yazılım test ekip üyeleri ile yapılacak çalışma sonucunda belirlenerek liste haline getirilir [Tablo 2] ve aday test araçlarının bu kıstasları sağlayıp sağlamadığı belirlenir. Yazılım test otomasyonu için ayrılan bütçe de önemli bir kıstastır fakat ön araştırma yapılırken daha çok test aracının yeteneklerine odaklanmak ihtiyacı karşılayacak test aracını belirlemede daha doğru karar verilmesini sağlayacaktır.

Tablo 2 Test aracı değerlendirme kartı

	KISTASLAR						
1	**Uygunluk**	**Puan (0-10)**	**Ağırlık (1-10)**	**Puan* Ağırlık**	**Üye 1**	**Üye 2**	**Üye n**
1.1	Masaüstü uygulama desteği (.NET, Java vb.)						
1.2	Windows 32-bit ve 64-bit desteği						
1.3	Windows dışındaki diğer işletim sistemleri desteği						
1.4	"Data-driven" test geliştirme desteği						
1.5	"Recorded" test geliştirme desteği						
1.6	"Scripted" test geliştirme desteği						
1.7	"Debug-Breakpoint" desteği						
1.8	Karşılaştırma (dosya, görüntü, tablo) yeteneği						
1.9	"Conditional check point" yeteneği						
1.10	"Performance test" yeteneği						
1.11	"Load test" yeteneği						
1.12	"Explaratory test" yeteneği						
1.13	"Manuel Test" sonuçlarını test raporuna ekleme özelliği						
1.14	"Dynamic Test List" yeteneği						
1.15	"DOORS" plug-in desteği						
1.16	"JIRA" plug-in desteği						
1.17	"JENKINS" plug-in desteği						
2	**Bilgiye Erişebilirlik**						
2.1	"Email notification" yeteneği						
2.2	Test Raporunun açık, okunabilir ve						

	özelleştirilebilir olması						
2.3	"Scheduling server" yeteneği						
2.4	Hazırlanan test tanımlarının belirli bir formatta ihraç edilebilmesi						
2.5	Test geçmişi bilgisi tutabilme yeteneği						
3	**Esneklik ve Genişletilebilirlik**						
3.1	"Test koşturma yardımcısı" yeteneği						
3.2	"Konfigürasyon kontrol" plug-in desteği						
3.3	Test tanımları tekrar kullanılabilirlik desteği						
3.4	Kullanıcı tanımlı arayüz objelerini tanıma yeteneği						
4	**Bakım**						
4.1	Test aracına verilen destek						
5	**Adaptasyon Kolaylığı**						
5.1	Kolay kullanılabilirlik						
5.2	Test araç dokümantasyonu ve eğitim materyali yeterliliği						
5.3	Test tanımı hazırlama kolaylığı						
6	**Lisans ve Maliyet**						
6.1	Lisans çeşitliliği						

Test aracının seçimi için verilen örnekte [Tablo 5] "Kepner-Tregoe" yöntemi kullanılarak test aracı değerlendirmesi yapılmıştır. Seçim kıstaslarının belirlenmesinde genel kıstasların yanında seçilen projenin ve otomasyona dahil edilmesi düşünülen diğer projelerin özellikleri, organizasyonda kullanılan diğer yazılımlar, test ortamı vb. gibi daha özel kriterlerin eklenmesi seçim sonunda daha sağlıklı kararlar verilmesini kolaylaştıracaktır.

Kıstaslar belirlendikten sonra ekip üyeleri belirlenen kıstasların ağırlıklarını oylayarak kıstasların genel ağırlığını belirlerler. Genel ağırlık ekip üyelerinin verdiği ağırlık değerlerinin aritmetik ortalaması ile hesaplanır. Tablo 3'te üç kişilik değerlendirme ekibi ve test aracı değerlendirme kartının bir bölümü ile yapılan ağırlık hesaplaması verilmiştir. Sonraki adımda her ekip üyesi değerlendirdiği test aracı için belirlenen tüm kıstasları ayrı ayrı puanlar. Tablo 4'te üç kişilik değerlendirme ekibi ve test aracı değerlendirme kartının bir bölümü ile yapılan puan hesaplaması verilmiştir. Her bir kıstas için ekip üyelerinin verdiği ortalama ağırlık ve ortalama puan çarpılarak o kıstasa denk gelen değer hesaplanır. Tüm kıstaslar için bu işlem tekrarlanır. Elde edilen kıstas değerleri toplanarak değerlendirilen test aracının genel puanı hesaplanır. Bu değerlendirme aday her bir test aracı için tekrarlanır.

Tablo 3 Örnek test aracı ağırlık hesaplama kartı

	KISTASLAR				
3	**Esneklik ve Genişletilebilirlik**	**Ağırlık (1-10)**	**Üye 1**	**Üye 2**	**Üye 3**

3.1	"Test koşturma yardımcısı" yeteneği	**8**	*8*	*7*	*9*
3.2	"Konfigürasyon kontrol" plug-in desteği	**9**	*10*	*9*	*8*
3.3	Test tanımları tekrar kullanılabilirlik desteği	**7**	*8*	*7*	*6*
3.4	Kullanıcı tanımlı arayüz objelerini tanıma yeteneği	**10**	*10*	*10*	*10*

Tablo 4 Örnek test aracı puan hesaplama kartı

	KISTASLAR				
3	**Esneklik ve Genişletilebilirlik**	**Puan (1-10)**	**Üye 1**	**Üye 2**	**Üye 3**
3.1	"Test koşturma yardımcısı" yeteneği	**8**	*7*	*7*	*10*
3.2	"Konfigürasyon kontrol" plug-in desteği	**7**	*6*	*6*	*9*
3.3	Test tanımları tekrar kullanılabilirlik desteği	**8**	*7*	*10*	*7*
3.4	Kullanıcı tanımlı arayüz objelerini tanıma yeteneği	**7**	*8*	*6*	*7*

Tablo 5 Örnek test aracı değerlendirme kartı

	KISTASLAR			
3	**Esneklik ve Genişletilebilirlik**	**Ağırlık (1-10)**	**Puan (1-10)**	**Ağırlık* Puan**
3.1	"Test koşturma yardımcısı" yeteneği	**8**	**8**	*64*
3.2	"Konfigürasyon kontrol" plug-in desteği	**9**	**7**	*63*
3.3	Test tanımları tekrar kullanılabilirlik desteği	**7**	**8**	*56*
3.4	Kullanıcı tanımlı arayüz objelerini tanıma yeteneği	**10**	**7**	*70*
			Toplam	*253*

Genel puanlamada en yüksek puanı alan test araçları yazılım test ekip üyeleri tarafından ortak bir takım test senaryoları seçilerek deneme amaçlı kullanılır. Amaç kontrol listesi üzerinden yapılan değerlendirmeden sonra uygulamada çıkabilecek problemleri nihai kararı vermeden önce tespit etmektir. Deneme amaçlı kullanılması düşünülen test aracı sayısı ekibin büyüklüğü ve takvim genişliği doğrultusunda arttırılabilir. Ücretli uygulamaların birçoğu bu sebepten ücretsiz sınırlı süreli lisans sağlamaktadır. Süreli lisans kullanılarak ücretsiz uygulamaların yanında ücretli uygulamalar da denenmelidir.

Deneme süreci sonunda yazılım test ekip üyelerinden gelen geri bildirimler doğrultusunda belirlenen kıstasların gerçekten sağlandığına karar verilebilir. Test ekip üyelerinin bu süreçteki deneyimlerini ekibin diğer üyeleri ile paylaşması karar vermeyi kolaylaştırabilir. Böylece yazılım test ekibinin üzerinde hemfikir olduğu bir

test aracı seçilir. Seçilen test aracı, belirlenen seçim kıstasları ve sonuçlarını içeren bir rapor hazırlanarak üst yönetimin onayına sunulur. Seçilen test aracı ücretli bir uygulama ise yönetim onayı sonrasında uygun lisans için satın alımı iş emri verilir.

6 Alt Yapıların Otomasyona Uyarlanması

Yazılım test otomasyonu için kullanılacak araç seçimi ve varsa lisanslama yapıldıktan sonra süreçte önceden belirlenmiş olan otomasyona geçirilecek yazılım için hazırlanmış ya da hazırlanacak dokümanların şablonları otomasyona uygun hale getirilmelidir. Bu aşamada yazılım gereksinim özelliklerini belirten dokümanlardan test tanımlarının yapıldığı dokümanlara kadar bütün dokümanlar taranıp test aracının seçiminde elde edilen bilgiler kullanılarak test otomasyonuna hazır hale getirilmelidir. Özellikle manuel test için hazırlanan test tanımları ve test raporları bu süreç boyunca değişikliğe ihtiyaç duyacaktır. Örneğin test otomasyon aracında oluşturulan senaryolar yazılım test tanımı dokümanı olarak ve testler koşturulduktan sonra oluşan test aracının çıktıları da test raporu dokümanı olarak kullanılabilir. Böylece konfigürasyon kontrolü altına alınan yazılımın ilgili dokümanlarını(test tanımları dokümanı, test raporu, test girdileri vb.) tekrar hazırlamak için işgücü kaybının önüne geçilmiş olur.

Ayrıca test aracının yetenekleri de göz önünde bulundurularak bazı senaryolardaki işlem yüklerinin kullanılan test yazılımına aktarılması ile test otomasyon aracı ve test yazılımı arasında yapılacak işlemlerin paylaştırılması söz konusu olabilir. Bu tip durumlar da değerlendirilerek testlerin otomasyona aktarılma süresini kısaltmak için test yazılımlarına ilave yetenekler eklenmelidir. Örneğin bir test senaryosunun koşturulması için test otomasyon aracı üzerinde kodlama yapılması gerekiyorsa ve o senaryonun test yazılımında kodlanması daha az işgücü gerektiriyorsa tercih test yazılımına senaryonun kodlanması yönünde olmalıdır.

7 Testlerin Otomasyon Aracına Aktarılması

Hazırlanan ya da daha önce hazırlanmış olan test tanımlarının test aracına aktarılması işlemidir. Daha önce ön çalışması yapılarak tecrübeler kazanılmış test aracı ile ekip olarak çalışmalara başlanır. Testlerin tamamının aktarılması için planlama yapılmamalı, her sürümde koşturulacağı düşünülen testlere öncelik verilmelidir. Ayrıca yine karmaşık senaryoları olan testlerden başlamak yerine daha basit ve tekrarlı testleri ilk başlarda aktararak hem ekibin test aracını ve test metodolojilerini öğrenmesi sağlanır aracına olan yabancılığı giderilir hem de testlerin hazırlanması ve koşturulması planlanan tarihlerden sapmadığı için ekibin motivasyon kaybı yaşamaması sağlanır.

Özellikle kullanıcı arayüzü yazılımlarının testlerinin otomasyona aktarılmasında unutulmaması gereken noktalardan bir tanesi, test otomasyonunun hiç bir zaman manuel testlerin yerini almayacağıdır. Test otomasyonu son kullanıcının uygulamayı kullanırken yaşadığı gerçek deneyimin, kullanım kolaylığı gibi müşteri memnuniyetini etkileyebilecek detayların ölçülmesinde kullanılamaz. Bu sebeple test

otomasyonu yanında manuel testler de kaliteyi doğrudan ilgilendirdiği için gereklidir ve yapılmalıdır.

Ayrıca testler esnasında kullanıcı etkileşimli işlemler yapılacaksa bu gibi testlerin de test aracına aktarılmaması gerekir. Örneğin yazıcı çıktısının karşılaştırıldığı bir testi ya da harici bellek takılması/çıkarılması işlemlerinin yapıldığı testler ya da bazı donanımların fiziksel olarak çıkarılıp takılması ile ilgili olan testler otomasyon aracına aktarılmamalıdır.

8 Test Otomasyonuna Geçirilen Testlerin Konfigürasyon Kontrolü Altına Alınması

Testler, test otomasyon aracına aktarıldıktan sonra mutlaka konfigürasyon kontrolü altına alınmalıdır. Geliştirilen testler dışında test edilen yazılım, yazılım gerekleri, test tanımları ve oluşturulan dokümanların konfigürasyon kontrolü altına alınması versiyondan versiyona yapılan değişikliklerin ölçülebilmesini ve versiyon kontrolü altına alınan test tanımlarının gerektiğinde tekrardan koşturulabilmesini sağlar.

Versiyon kontrolü özellikle test konfigürasyonunun korunmasını sağlamaktadır. Test edilen yazılım ile test aracında oluşturulmuş test dosyalarının aynı konfigürasyon kontrol aracı altında tutulması ile yazılımın istenen sürümü için tekrar test yapılması istendiğinde ilgili yazılım sürümüne karşılık gelen test aracı dosyalarını kullanarak testler tekrar edilebilir.

9 Yazılım Test Otomasyonu ve Sürekli Entegrasyon

Sürekli entegrasyon bir yazılım geliştirme ekibinin geliştirdiği kod parçacıklarının sürekli ve belirli bir sıklıkla birleştirilmesiyle gerçekleştirilen bir yazılım geliştirme yöntemidir. Çevik yöntemler genellikle bir veya iki haftalık kısa iterasyonlara dayanır. Sürekli entegrasyon bu tekrarlarda koşturulan otomatik testler ile sonuçları hızlıca yazılım geliştirme ekibine bildirebilir. Sürekli entegrasyona uygun şekilde tasarlanan otomatik testler genelde yazılımın test edilmesi gereken kısmının büyük bir bölümünü kapsar. Böylece her bir tekrarlamada yazılıma eklenen/değiştirilen kod parçacığının etkisi kaliteden ödün verilmeden, manuel test için harcanan sürenin çok altında bir sürede gözlemlenebilir ve ilgili paydaşlara raporlanabilir.

Yazılım test otomasyonu sürekli entegrasyona dahil edilmiş bir yazılım projesinde kullanıldığında maksimum faydayı sağlayabilecektir. Ayrıca yapılan test tekrarları sağlanan faydayı arttıracaktır.

10 Test Sürecinde Ölçüm

Testlerin manuel ya da otomatik olarak yapılmasından bağımsız olarak test süreçlerinin ölçülmesinde test metrikleri kullanılmalıdır. Test metrikleri ile yapılan ölçümler, otomasyona aktarılan testlerin bakım sürecinde ne gibi güncellemeler

yapılması gerektiği hakkında bizlere bilgi verecektir. Bu bilgiler ışığında testlere yeni senaryolar eklenmeli, bir sonraki koşumlarda tekrar ölçümler yapılarak sürekli iyileşme sağlanmalıdır. Ayrıca test metrikleri, test edilen yazılımın hedeflenen kalite seviyesine ne kadar ulaştığını ve yazılımın dağıtıma hazır olup olmadığını belirlemeye yarayan objektif ölçümlerdir.

Temel test metrikleri;

- **Bulunan Sorun Sayısı**: Test verimliliğini ölçmeye yarayan metriklerden biridir. Ancak dikkat edilmesi gereken iki husus vardır:

  a. Yazılımda mevcut olan sorun sayısı (bulunmuş ya da bulunmamış) "Bulunan Hata Sayısı" nı etkiler.

  b. Tüm sorunlar eşit önemde değildir, sorunlara "önem seviyesi" verilerek doğru yazılım kalite ölçümü yapılır.

- **Sorun Bulma Verimliliği (SBV):** Test verimliliğini ölçmeye yarayan kuvvetli bir metriktir.

  SBV = A / (A + B)

  A = Testler sırasında bulunan sorun sayısı

  B = Son Kullanıcı tarafından bulunan sorun sayısı

  SBV'nin başarısı bazı faktörlere bağlıdır.

  a. Bulunan hataların önem derecesi dikkate alınmalıdır.

  b. Son Kullanıcının etkisi önemlidir.

- **Sorun Yaşı ve Bozulmuşluk:** Sorunu yazılım geliştirme sürecinin erken safhalarında bulmak çok önemlidir. Sorun Yaşı ve Bozulmuşluk metrikleri aynı temel ile çalışır, hatanın ne kadar geç bulunduğunu bulmaya yarar. Sorun Yaşı aşağıdaki gibi bulunur [Tablo 6]:

  Bozulmuşluk = (Σ (A * B)) / Σ A

  A: Bir fazda bulunan hata sayısı

  B: Sorun Yaşı

Tablo 6 Sorun yaşı*

	Sorunun Bulunduğu Faz				
Sorunun Yaratıldığı Faz	Gereksinim Hazırlama	Tasarım	Geliştirme	Test	Müşteri Ortamı
Gereksinim Hazırlama	0	1	2	3	4
Tasarım	0	0	1	2	3
Geliştirme	0	0	0	1	2

*Hücrelerdeki sayılar yaşı ifade eder.

11 Sonuç

Yazılım test otomasyonuna geçme kararı vermeden önce ihtiyaçlar ve beklentiler belirlenmeli, yazılım test aracı olarak kullanılması düşünülen yazılım araştırılmalıdır. Planlama yaparken süreçte zaman ve iyi bir ekibe ihtiyaç duyulacağı unutulmamalıdır. Test otomasyonu sürekli gelişmeye açık bir süreç olduğundan yapılacak ölçümler ile eksik ve ya hatalı bulunan yönleri düzeltilebilir. Sürecin ilk başında test otomasyonuna karar verilmesi aşamasında yapılan maliyet ve fayda analizi süreç boyunca tekrarlanabilir ve her tekrarda artan tecrübe ve birikim ile gerçeğe daha yakın sonuçlar elde edilebilir.

Teşekkür

Bu makaleye tavsiyeleri ve yorumları ile katkı sağlayan Ergün Doğan'a katkılarından dolayı teşekkür ederiz.

Referanslar

1. Fewster, M., Graham, D.: Software Test Automation Effective Use of Test Execution Tools. Addison-Wesley, New York (1999)
2. Kaner, C., Bach, J., Pettichord, B.: Lessons Learned in Software Testing. Wiley Computer Publishing, New York (2002)
3. Sarıalioğlu, B.:Software Testing Tips Experiences and Realities. Barış Sarıalioğlu, İstanbul (2014)
4. Gürbüz, A.: Yazılım Test Mühendisliği. Papatya Yayıncılık Eğitim, İstanbul (2010)
5. Ramler, R., Wolfmaier, K.:Economic Perspective in Test Automation: Balancing Automated and Manual Testing with Oppurtunity Cost. Austria, pp 85-91
6. Hayes, L., G.:The Automated Testing Handbook. 1996

7. ISTQB Foundation Level Syllabus, 2011