OTOMASYON TESTLERİNİN YAZILIM GELİŞTİRME SÜRECİNE ETKİSİ, MANUEL VE OTOMASYON TESTLERİNİN KARŞILAŞTIRILMASI

Beyzanur AKÇELİK
20 14 19 220

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Tezsiz Yüksek Lisans Programı
Danışman: Dr. Öğr. Üyesi Erdal GÜVENOĞLU

İstanbul
T.C. Maltepe Üniversitesi
Lisansüstü Eğitim Enstitüsü
Haziran 2022

OTOMASYON TESTLERİNİN YAZILIM GELİŞTİRME SÜRECİNE ETKİSİ, MANUEL VE OTOMASYON TESTLERİNİN KARŞILAŞTIRILMASI

Beyzanur AKÇELİK
20 14 19 220
ORCID: 0000-0002-3639-959X

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Tezsiz Yüksek Lisans Programı
Danışman: Dr. Öğr. Üyesi Erdal GÜVENOĞLU

İstanbul
T.C. Maltepe Üniversitesi
Lisansüstü Eğitim Enstitüsü
Haziran 2022



ETİK İLKE VE KURALLARA UYUM BEYANI

Bu belge, Yükseköğretim Kurulu tarafından 19.01.2021 tarihli *"Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge"* ile bildirilen 6689 Sayılı Kişisel Verilerin Korunması Kanunu kapsamında gizlenmiştir.

TEŞEKKÜR

Tez çalışmam süresince gerekli danışmanlığı ve yardımlarını eksik etmeyen tez danışmanım Dr. Öğr. Üyesi Erdal GÜVENOĞLU başta olmak üzere, manuel test ve otomasyon testlerine ait test senaryolarını referans aldığım Turkuvaz Medya Grubu şirketine, yazılım test üzerine akademik çalışmalar yapan tüm mühendis meslektaşlarıma ve hayatımın her döneminde olduğu gibi yüksek lisans eğitimim boyunca da destek olan aileme sonsuz teşekkürlerimi sunarım.

Beyzanur AKÇELİK

Haziran 2022

ÖZ

OTOMASYON TESTLERİNİN YAZILIM GELİŞTİRME SÜRECİNE ETKİSİ, MANUEL VE OTOMASYON TESTLERİNİN KARŞILAŞTIRILMASI

Beyzanur AKÇELİK
Yüksek Lisans Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Tezsiz Yüksek Lisans Programı
Danışman: Dr. Öğr. Üyesi Erdal GÜVENOĞLU
Maltepe Üniversitesi Lisansüstü Eğitim Enstitüsü, 2022

Her geçen gün artan müşteri talepleri sektörleri dijitalleşmeye mecbur kılmaktadır. Yazılım sektörü kendi içerisinde dijitalleşmiş görünse de halen dijitalleşme dönüşümü gereken alanları vardır. Bu sebeple, başarılı yazılımların geliştirilmesi için bu konudaki standartların ve belirlenmiş süreçlerin en iyi şekilde uygulanması büyük önem arz etmektedir. Projelerin daha kaliteli ve başarılı olması için yazılım kalitesi ve yazılım testi büyük önem taşımaktadır. Müşteri taleplerinin karşılanması ve memnuniyetinin artırılması yazılım test ve kalite faaliyetleri ile doğrudan ilişkilidir.

Bu çalışma kapsamında, yazılım sürecinin içerisinde yer alan test adımı en az yazılımın geliştirilmesi kadar önemlidir. Yazılım projelerindeki dokümantasyon, test senaryoları, performans, kullanılabilirlik, kalite faaliyetleri gibi etkenler yazılım testinin sonucunu etkilemektedir. Bunları değerlendirdiğimizde yazılım testinin güvenli ve hızlı olması ne denli önemli olduğu ortaya konulmaktadır. Proje genelinde zaman, kaynak ve para tasarrufu sağlayan yazılım testlerinin çeşitlerinden bahsedilmektedir.

Anahtar Sözcükler: Yazılım testi, Yazılım Performansı, Test Otomasyon, Manuel Test, Yazılım kalitesi

ABSTRACT

EFFECT OF AUTOMATION TESTS ON SOFTWARE DEVELOPMENT PROCESS, COMPARISON OF MANUAL AND AUTOMATION TESTS

Beyzanur AKÇELİK
Master Thesis
Computer Engineering
Computer Engineering (Without Thesis)
Thesis Advisor: Assist. Prof. Dr. Erdal GÜVENOĞLU
Maltepe University Graduate School, 2022

Increasing customer demands are forcing the industries to digitize. Although the software industry seems to be digitized in itself, there are still areas that need digitalization transformation. For this reason, it is of great importance to apply the standards and determined processes in the best way for the development of successful software. Software quality and software testing are of great importance for projects to be of higher quality and success. Meeting customer demands and increasing satisfaction are directly related to software testing and quality activities.

Within the scope of this study, the testing step in the software process is at least as important as the development of the software. Factors such as documentation, test scenarios, performance, usability, quality activities in software projects affect the result of software testing. When we evaluate them, it is revealed how important it is for software testing to be safe and fast. Various types of software tests that save time, resources and money are mentioned throughout the project.

Keywords: Software Testing, Software Performance, Test Automation, Manual Test, Software Quality

İÇİNDEKİLER

ETİK İLKE VE KURALLARA UYUM BEYANI ii
TEŞEKKÜR iii
ÖZ iv
ABSTRACT v
İÇİNDEKİLER vi
TABLOLAR LİSTESİ vii
ŞEKİLLER LİSTESİ viii
KISALTMALAR ix
ÖZGEÇMİŞ x
BÖLÜM 1. GİRİŞ 1
1.1. Problem 1
1.2. Amaç 1
1.3. Önem 2
1.4. Varsayımlar 2
1.5. Sınırlılıklar 2
1.6. Tanımlar 3
BÖLÜM 2. LİTERATÜR ARAŞTIRMASI 5
BÖLÜM 3. YÖNTEM 11
3.1. Araştırma Modeli 11
3.2. Evren 11
3.3. Örnekleme 11
3.4. Veriler ve Toplanması 12
3.5. Verilerin Yorumlanması 14
3.6. Süre ve Maliyet 14
BÖLÜM 4. BULGULAR VE YORUMLAR 15
4.1. Bulgular 15
4.2. Yorumlar 16
BÖLÜM 5. SONUÇ 17
KAYNAKÇA 18

TABLOLAR LİSTESİ

Tablo 1.1. Yazılım Test Uygulamaları Konusunda Geçmiş Anketler 7

ŞEKİLLER LİSTESİ

Şekil 3.1 Seçili Özelliğin Süreç Akışı 13

KISALTMALAR

PROD : Production

DL : Deadline

UAT : User Acceptance Testing

UZM : Uzman

SOA : Service-Oriented Architecture

BT : Bilişim Teknolojileri

ÖZGEÇMİŞ

Beyzanur AKÇELİK

Bilgisayar Mühendisliği Anabilim Dalı

Eğitim

Derece	*Yıl*	*Üniversite, Enstitü, Anabilim/Anasanat Dalı*
Y.Ls.	2022	Maltepe Üniversitesi Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı
Ls.	2020	İnönü Üniversitesi, Mühendislik Fakültesi Bilgisayar Mühendisliği
Lise	2016	Taksim Teşvikiye Lisesi

İş/İstihdam

Yıl	*Görev*
2020 -	Analist / Test Uzmanı. Barsan Global Lojistik
2021-	Test Uzmanı. Turkuvaz Medya Grubu

Mesleki Birlik/Dernek Üyelikleri

Yıl	*Kurum*

Alınan Burs ve Ödüller

Yıl	*Burs/Ödül*

Yayınlar ve Diğer Bilimsel/Sanatsal Faaliyetler

BÖLÜM 1. GİRİŞ

Yazılım testleri, üzerinde çalışılan bir yazılım sisteminin kontrolünün yapılarak ihtiyaçların ve istenenlerin tam olarak karşılanması için var olan bir işlemler dizisidir. Test işlemleri kara kutu testlerinden beyaz kutu testlerine, ünite testlerinden sistem testlerine, birleştirme testlerinden uygulama testlerine kadar birçok test adımını kapsamaktadır [4]. Bu bölümde, araştırmanın kimlik bilgilerinden olan problem, amaç, önem, varsayımlar, sınırlılıklar ve tanımlara yer verilmiştir.

1.1. Problem

Bir yazılım en hızlı ve güvenilir şekilde test edilmesi günümüzde en büyük sorunlardan biridir. Günümüzdeki tüm yazılımsal ürünlere kullanıcıları tarafından istekler gelir. Bu istekleri bir an önce tamamlayıp prod ortamı dediğimiz son kullanıcıya sunma süresinde en fazla zaman geçen süre test süresidir. Geliştirilen bir yazılım, test edilmeden prod ortamına alınamayacağı için DL denilen sürenin uzun olması test senaryolarının detaylı şekilde denenmesinden kaynaklıdır. Talep sahibi isteğinin hemen olmasını ister. Keza yazılım geliştirici ise yaptığı işin hemen test edilip revizesi varsa hızlıca düzenlemek ister. Bu küçük örneklerde dahi yazılım test süreci ciddi rol almaktadır. Yazılım test eden kişi için de süresel sıkıntı vardır. Bir yazılım testi için harcanan süreyi kısaltabilirsek diğer yazılım testlerine ayrılan süreyi arttırabilir ve böylelikle olası sorunların tespiti kaçırılmaz. Problemimiz, bir yazılım testinin uzun sürmesi ve olası hataların öne geçilememesidir.

1.2. Amaç

Araştırmanın amacı, yazılım testlerinin en güvenli ve hızlı şekilde yapabilmektir. Yazılım testinin güvenli şekilde çıkması demek son kullanıcı kullanırken karşılaşma ihtimali olan sorunların önüne geçilmesidir. Günümüzde teknoloji ile çok yakın olan Z kuşağı herhangi bir uygulamayı kullanırken karşılarına çıkan hatalara tahammül etmeyip ilgili ürünü kullanmaktan vazgeçmektedir. Bu da ürünün marka ve prestij bakımından değer kaybetmesi demektir. Yazılım testi yapılırken denenen senaryoların tespiti çok kritiktir. Yazılım testi ilgili ürünün marka ve prestijine çok büyük katkı sağlamaktadır.

1.3. Önem

Yazılım testleri sonucunda yazılımdaki olası hataların tespiti konusunda en büyük rol oynayan süreçtir. Belirlenen senaryoların ilgili yazılımda yürütülmesi ve sonuca ulaşması ne kadar uzun sürerse ilgili talep müşteriye o kadar geç ulaşır. Bu süreyi en optimum hale getirirken ama süreden kısarken güvenlikten de ödün vermemek gerekir.

Bir yazılım güvenliği içeride yapılan transcation (işlem) türlerine göre değişmektedir. Örnek olarak bankacılık sektörü verilebilir. 2022 yılının mayıs-haziran döneminde Türkiye de yaşanan bir yazılımsal hata sonucu bir bankanın mobil uygulamasından müşterinin hesabına ciddi para transferi olmuştur. Burada son kullanıcı sadece mobil uygulamada bir kaç tuş kombinasyonuna basarak normal işlem yaptığından bir hacking işlemi yoktur. Burada tamamen yazılımın test sürecindeki gözden kaçan bir senaryo sonucudur.

1.4. Varsayımlar

Bir yazılımın tamamlanması ve son kullanıcıya sunulmasına kadar ki süreçte en önmeli adım yazılım testidir. Bu testler de kullanılan senaryolar da bu adımdaki en önemli kısımdır. Otomasyon veya manuel olarak test edilen yazılım tüm senaryolardan geçmeden son kullanıcıya sunulamaz. Mümkün olduğu sürede test süreçleri otomasyon halinde yapılması daha uygundur.

1.5. Sınırlılıklar

Bir yazılımın test edilmesinde en önemli zorluk ilgili yazılımın kurgusu konusunda dokümantasyonun olmamasıdır. Literatür de FDD denilen yazılımın operasyon süreci hakkında bilgi dokümanıdır. Bu bilgi dokümanı ilgili sürecin akışını ve teknik açıdan nasıl çalıştığı ile ilgili bilgiler barındırmaktadır. Uzun süredir geliştirilen bazı yazılımlarda doküman bulmak şu anda en büyük sorundur. Bu durumlarda manuel test yapmak zorunda kalınmaktadır.

1.6. Tanımlar

Yazılım Testi: gerçekleştirilen bir talebin kabul kriterlerine göre istenilen yapılıp yapılmadığını ilgili yazılımı kullanarak kontrol edilmesidir. Bu süreç temel olarak yazılım kalitesini arttırılmasını hedeflemektedir. Bu süreç özünde iki parçadan oluşmaktadır:

Doğrulama: Manuel ve kaşif testininin en etkili yer olduğu doğrulama süreci, test edilen yazılımın müşterinin istediği gereksinimleri karşılayıp karşılamadığına odaklanır. Yani uygulama hatasız, mükemmel çalışıyor olabilir fakat müşterinin belirttiği şekilde çalışmıyor ise eğer bu testi geçemez. Daha sonlara doğru yapılan bir aktivitedir.
Amaç: İnsanın sahip olduğu tecrübe, zeka, gözlem, duygu ve kullanılabilirlik açısından uygulamayı test etmektir.
Geçerleme: Yazılım geliştirme döngüsünün her safhasında gerçekleşir. Uygulamanın hatasız, doğru şekilde çalışıp çalışmadığına odaklanır. Test otomasyonunun aktif olarak kullanıldığı alan da burasıdır.
Yazılım Testinin Amaçları

- Müşteriye sunmadan istenilenin yapıldığına emin olmak,
- Daha sonradan hatalı olarak talebin geri gelmesi riskini azaltmak,
- Hataları erken teşhis ederek yazılım maliyetini azaltmak,
- Yazılım geliştiricinin odak noktası çok değişmeden tekrar düzenlemesini istemek,
- Müşteri memnuniyetini arttırmak ve izleyen siparişler için zemin hazırlamak.

Manuel Test: Manuel test, test senaryolarını otomatik bir araç kullanılmadan manuel olarak yürütüldüğü bir yazılım test sürecidir. Tüm test senaryoları test cihazı tarafından son kullanıcı düşünülerek manuel olarak yürütülür. Test senaryosu raporları da manuel olarak oluşturulur. Manuel Test, yazılımın hem görünür hem de gizli hatalarını bulabildiği için en temel test süreçlerinden biridir.

Test Otomasyonu: Yazılım ürünleri üzerinde yapılan test sürecinin araçlar kullanılmasını içeren süreci ifade etmektedir. Testlerin çalıştırılmasını, verilerini yönetilmesini ve elde edilen sonuçların yorumlanmasını barındırmaktadır. Her yazılım test otomasyon sistemlerine uygun olmayabilmektedir.

Yazılım Test Süreci: Test uzmanları, test ettiği bir sürecin beklendiği gibi bir sonuç verip vermediğini kontrol etmektedir. Doğru sonucu vermez ise ilgili hataları tespit eder ve çözülmesi için yazılım departmanına bildirilecektir. İzlenen bu yol sayesinde yazılım kalitesi de artmaktadır. Ürün teslimi gerçekleştiğinde bile yeni özellik talepleri olacaktır. Bu yeni özelliklerinde test edilmesi mutlaka gerekecektir. Yazılım testi asla bitmeyen bir süreç olup sadece ürün tesliminde değil, her zaman kendini göstermektedir.

BÖLÜM 2. LİTERATÜR ARAŞTIRMASI

Yazılım testlerinde halen manuel test kullanan şirketler mevcut. Bunun ana sebebi ise ilgili süreçlerin otomatik hale getirilmesinin mümkün olmamasıdır. Örnek bir sektör verilmesi gerekirse lojistik sektörünü verebiliriz. Geliştirilen sürecin otomasyonel olarak mesela selenium ile test etmek mümkün değildir. Çünkü süreç fiziksel olarak depoda barkod okutma el terminalleri veya tablet üzerinden ilerlemektedir. Kullanılan servisleri otomasyon sistemi çağırabilir ama sürecin baştan uca sağlıklı ilerlemesi adına ve UAT denilen kullanıcı kabul testinden geçmesi için manuel depoda testin gerçekleşmesi istenmektedir.

Yazılım test sektörünün kritik hale gelmesinin başlıca sebeplerinden biri de dijitalleşme ve mobil kullanım artmasıyla yazılımın girmediği bir alan kalmamaktadır. Her yıl 1/5'i kadar hızla gelişen sektör, yazılım alanında yeni geçiş sağlamış olan sektörler sayesinde bu hız gittikçe artmaktadır. Önceden test süreçleri manuel olarak yürütülmekteydi. Günümüzde ve yakın geçmişte test otomasyon sistemleri manuel testlerin yerini almaktadır. Yazılımsal geliştirmenin her alanda bu kadar fazla olması son kullanıcıların hataya, yavaş çalışmaya tahammülleri olmamaktadır [URL-1].

Yazılım testinde hizmet veren Testinium şirketi günümüzde çok fazla yazılım ürününe yazılım testi hizmeti vermektedir. Bir yazılım ürünündeki talebin en hızlı ve güvenli şekilde test edilmesini amaçlamaktadır. Şirket CEO'su Melih SAKARYA şu sözleri ile test konusunun önemini dile getirmektedir [URL-1].

"DevOps, dijital dönüşüm projeleri ve çevik proje yönetimi bunun itici gücü. Yaşanacak bir hata, operasyonel, regülasyon ve prestij anlamında ciddi kayıplara neden olabilir"

Her alanda gelişen yazılım ürünlerinin vazgeçilmez bir parçası olan yazılım kalitesi ve test faaliyetleri günümüzde önemini açık bir şekilde göstermektedir [URL-1]. Virgosol Kurucu Ortağı Miraç Emektar'a göre;

"Yapay zeka, makine öğrenmesi, nesnelerin interneti ve büyük veri gibi teknolojilerin yaygınlaşması, farklı seviye ve çeşitteki yazılım test ihtiyacını da getiriyor."

Günümüzde Türkiye'de gerçekleşen testlerin ilk sırasını fonksiyonel test türü almaktadır. Her alanda yazılımın bulunması güvenlik konusunda ciddi endişeler doğurmaktadır. Bu yüzden güvenlik testlerini önemi artmaktadır [URL-1].

Yazılım test uygulamaları konusunda değişik ülke ve büyüklüklerde yüksek sayıda anket yapılmıştır. Yapılan literatür araştırması sonucunda geçmişte yapılan çalışmalardan bazılarını aşağıda görebilirsiniz.

Tablo 1.1. Yazılım Test Uygulamaları Konusunda Geçmiş Anketler

Büyüklük / bölge	**Yıl**	**Katılım sayısı**	**Hedef/Odak alanı**
ABD	1983	207	Test prosedürleri
Washington, ABD	1988	-	Test metodolojileri, test süreç modelleri, profesyonelliğin büyümesi
Hindistan	2002	-	Test standartları/prosedürleri, donanım platformu, test uygulaması, eğitim, ölçütler, test sertifikasyonu
İsveç	2003	11	İsveç şirketi Test süreçleri
İsveç	2003	91	Genel yazılım testi (ve yeniden kullanım)
Kanada	2004	60	Test uygulamaları, test türleri/seviyeleri, teknikleri, otomasyonu, ölçütleri, yönetimi, eğitimi
Avustralya	2004	65	Test teknikleri, araçları, ölçütleri, standartları
Finlandiya	2005	18 uzm	Testte araştırma istikametlerinin sıralanması
Avustralya	2006	35	Doğrulama ve geçerli kılma bağlamı ve kullanımı karar süreci, enformasyon ve maliyet etkinlik vaka çalışması
İsveç	2006	17	Birim testi
İsveç	2006	12 şirket	Test olgunluğu
Birleşik Krallık	2007	1 Şirket	Entegrasyon testi, gereksinim testi, test kapsama, test otomasyon, test senaryo tasarımı
Alberta eyaleti, Kanada	2009	53	Test uygulamaları, test türleri/seviyeleri, teknikleri, otomasyon, araçlar, ölçütler, yönetim, eğitim
Çevrimiçi topluluk	2009	121	Test Güdümlü Geliştirme'nin benimsenme oranı ve Test Güdümlü Geliştirme ile ilişkili teknikler
İsveç	2010	32	Regresyon testi
Avrupa	2010	83	Yazılım testinin çağdaş görünümü
Kanada	2010	246	Test uygulamaları, test türleri, teknikleri, araçlar, ölçütler, yönetim, akademi ile araştırma ve etkileşim
Türkiye	2013	-	Test teknikleri, test yeterliği, test amacı, çıkış kriteri, bulut testi, mobil test, olgunluk ve standardizasyon, keşif testi, yetenek kümesi

2.1. Geçmiş Güvenlik Testi Çalışmaları

Günümüz yazılım sistemlerinin boyutu ve karmaşıklığı büyüyor, bu da hata sayısını ve dolayısıyla güvenlik açıkları olasılığını artırıyor [5]. Yazılım testi, bir programın yapmak üzere tasarlandığı şeyi yapmadaki başarısızlığını oldukça kolay bir şekilde algılar, ancak süreçte yapabileceği diğer tüm şeyleri algılamaz. Bunlardan bazıları güvenliği tehlikeye atabilir. Makale, bu hatalarla başa çıkmak için yöntemlerin ve yazılım araçlarının geliştirilmesini tartışıyor [6].

Yazılım geliştiriciler, hatalı yazılım üretildiğinde ve güvenlik açıklarına neden olan kusurlar keşfedildiğinde neyin tehlikede olduğunu tam olarak kavrayamazlar. Bu sorunu çözmek için, yazılım güvenlik açığı süreçlerine dahil olan ana aktörler ve bu gruplar içindeki ilgili roller belirlenir [7]. Gelişmekte olan yazılım güvenliği alanının karşılaştığı gerçek zorluklardan biri, kolayca erişilebilir ortak bir bilgi birikiminin olmamasıdır [8].

Mevcut yazılım güvenliği teknikleri, giderek daha fazla birbirine bağlanan toplumumuzun talep ettiği güvenli sistemleri üretemiyor, bu nedenle daha etkili ve ölçeklenebilir bir yaklaşıma olan dinamik yazılım güvenlik testine ihtiyaç duyuyor [9]. Savunma ve ulusal güvenlik yazılımlarının çalışması gereken ortamın mevcut durumuna genel bir bakış sağlar, ardından yazılım güvenliği güvencesinin çeşitli yönlerini teşvik etmeye dahil olan mevcut ve ortaya çıkan faaliyetleri ve kuruluşları araştırır [10].

Güvenlik, yazılım geliştirmede ciddi bir sorundur ve gelecekte çok daha kötü hale gelebilir. Ne yazık ki yazılım güvenliği sorununun basit bir çözümü yoktur. Günümüzün yazılım geliştiricileri, piyasaya sürülen yazılımlardaki çok sayıda kusurla uğraşmak zorundadır [11]. Bu belge, kara kutu test araçları hakkındadır. Bu terimi, test edilen sistemin kara kutu görüntüsünü alan araçlara atıfta bulunmak için kullanıyoruz; yazılım kaynak kodunun veya mimarisinin kullanılabilirliğine güvenmezler ve genel olarak yazılımın davranışını dışarıdan keşfetmeye çalışırlar [12].

SOA güvenliği, bir yazılım programı veya platformu yerine tüm BT sisteminin hedeflerini veya hedeflerini uygulayan bir güvenlik türüdür. Bu makale, mantıklı bir SOA güvenliği elde etmek için kaçınmamız gereken 13 tuzak açıklamaktadır [13]. Güvenlik değerlendirmesindeki mevcut çabaların çoğu, bilinen güvenlik açıklarını aramayı içerir. Bilinen güvenlik açıklarının özelliklerini ve doğasını anlayarak süreç geliştirilebilir. Bu şekilde kazanılan bilgi, daha sonra benzer ancak henüz bilinmeyen güvenlik açıkları için yeni sistemleri sistematik olarak incelemek için bir çerçeve olarak kullanılabilecek uygun bir sınıflandırma içinde organize edilebilir. Bu makale, bilgisayar sistemlerinde saldırı ve güvenlik açıklarının sınıflandırmalarını geliştirmek için yapılan önemli çalışmaların kapsamlı bir araştırmasını sunar [14].

Son 10 yılda standart programlama hatalarına yönelik kod incelemesi, statik kod analizi ile büyük ölçüde otomatikleştirildi. Önümüzdeki 10 yıl boyunca, program analizindeki gelişmeler, verimli kısıtlama çözücüler ve güçlü bilgisayarlar sayesinde testleri otomatikleştirmede ve test oluşturmada benzer ilerlemeler görmeyi bekliyoruz [15]. Yazılım güvenliği test araçları ve metodolojileri şu anda bol miktarda bulunuyor ve soru artık güvenlik için "test edilip edilmeyeceği" değil, "nerede ve ne zaman test edileceği" ve "sonra ne?" gibi görünüyor. Bu yazıda, güvenlik testi literatürünün bir incelemesini sunuyoruz ve bir uygulamanın testi tamamlandıktan sonra döngüyü kapatan ve bir sonrakine faydalı girdi sağlayan, keşfedilen güvenlik açıklarının kuruluş içi bir deposundan yararlanan bir yazılım güvenlik testi şeması öneriyoruz [16].

Uygulamalardaki güvenlik açıkları ve başarılı saldırılar yoluyla bunların yaygın olarak kullanılması bugünlerde yaygın. Güvenlik açıklarını önlemek için uygulamaları test etmek, bu sorunu çözmek için önemli bir adımdır. Bu çalışmada, program güvenlik testi çalışmasını analiz etmek için yedi kriter belirledik. Bunlar; güvenlik açığı kapsamı, test senaryolarının kaynağı, test oluşturma yöntemi, test düzeyi, test senaryolarının ayrıntı düzeyi, test otomasyonu ve hedef uygulamalardır [17]. Kara kutu web uygulaması güvenlik açığı tarayıcıları, güvenlik açıkları için web uygulamalarını araştıran otomatik araçlardır. Tekniğin mevcut durumunu değerlendirmek için, önde gelen sekiz araca erişim sağladık ve bu tarayıcılar tarafından test edilen güvenlik açıklarının sınıfı, hedef güvenlik açıklarına karşı etkinlikleri ve hakkında bir çalışma gerçekleştirdik [18].

Güvenlik testi, yazılımın güvenlik gereksinimleri için testleri tanımlar. Güvenlik gereksinimleri işlevsel değildir ve bu nedenle, işlevsel olana kıyasla farklı bir test yöntemi gerektirir. Bu çalışma, ilk olarak mevcut güvenlik testi yaklaşımlarına genel bir bakış sunar ve buna dayalı olarak, risk ve otomatikleştirilmiş iki boyutta model tabanlı güvenlik testleri için yeni bir sınıflandırma geliştirir [19]. Yazılım açıkları, bilgisayar güvenlik sorununun temel nedenidir. İnsanların belirli bir yazılımda var olan güvenlik açıklarını nasıl hızla keşfedebilecekleri bilgi güvenliği alanının her zaman odak noktası olmuştur. Bu makale, statik analiz, Fuzzing, sızma testi dahil olmak üzere yazılım güvenlik açığı teknikleri hakkında araştırma yapmıştır [20].

Penetrasyon testi sürecinde, bir sistemi test etmenin basit bir yolu, birincil analiz ve resmi yöntemlerle yapılır, Çoğu güvenlik açığının çevre ile hatalı bir etkileşim nedeniyle tetiklendiği gözlemine dayanarak. Burada, olası güvenlik kusurları için web uygulaması ve veritabanı entegrasyon sistemini test etmek için bir yaklaşım açıklanmaktadır. Bu yaklaşım, sızma testi sürecinin karmaşık ve uzun prosedürünü bağlamayı amaçlayan dinamik bir modele dahil edilmelidir [21].Artan koruma ihtiyacı ısındıkça, müşterileri etkilemek için bir terminoloji telaşı atılıyor, ancak genellikle sadece onları karıştırıyor. Çoğu müşteri, çeşitli güvenlik testi yöntemleri arasındaki önemli farklılıkları bilmiyor veya anlamıyor ve çok fazla satıcı, mallarını ve hizmetlerini satmak için bu cehalete güveniyor [22].

BÖLÜM 3. YÖNTEM

Bu bölümde, araştırma modeli, evren ve örnekleme, veriler ve toplanması, verilerin çözümlenmesi ve yorumlanması ile süre ve maliyet bilgilerine yer verilmiştir.

3.1. Araştırma Modeli

Yapılacak olan yazılım testi için Turkuvaz Medya Grubu şirketindeki birimler ile görüşülüp kullandığı ürünlerin dokümanları incelendi. İncelenen ürünlerin ilgili birimlerinden yeni özellikler hakkında bilgi istenildi. Bu özelliklerle ilgili atanılan yazılım ekipleri ile irtibata geçildi ve bu taleplerin geliştirilme sürecinin ne zamanki planlarına alındığı hakkında görüşmeler sağlandı.

3.2. Evren

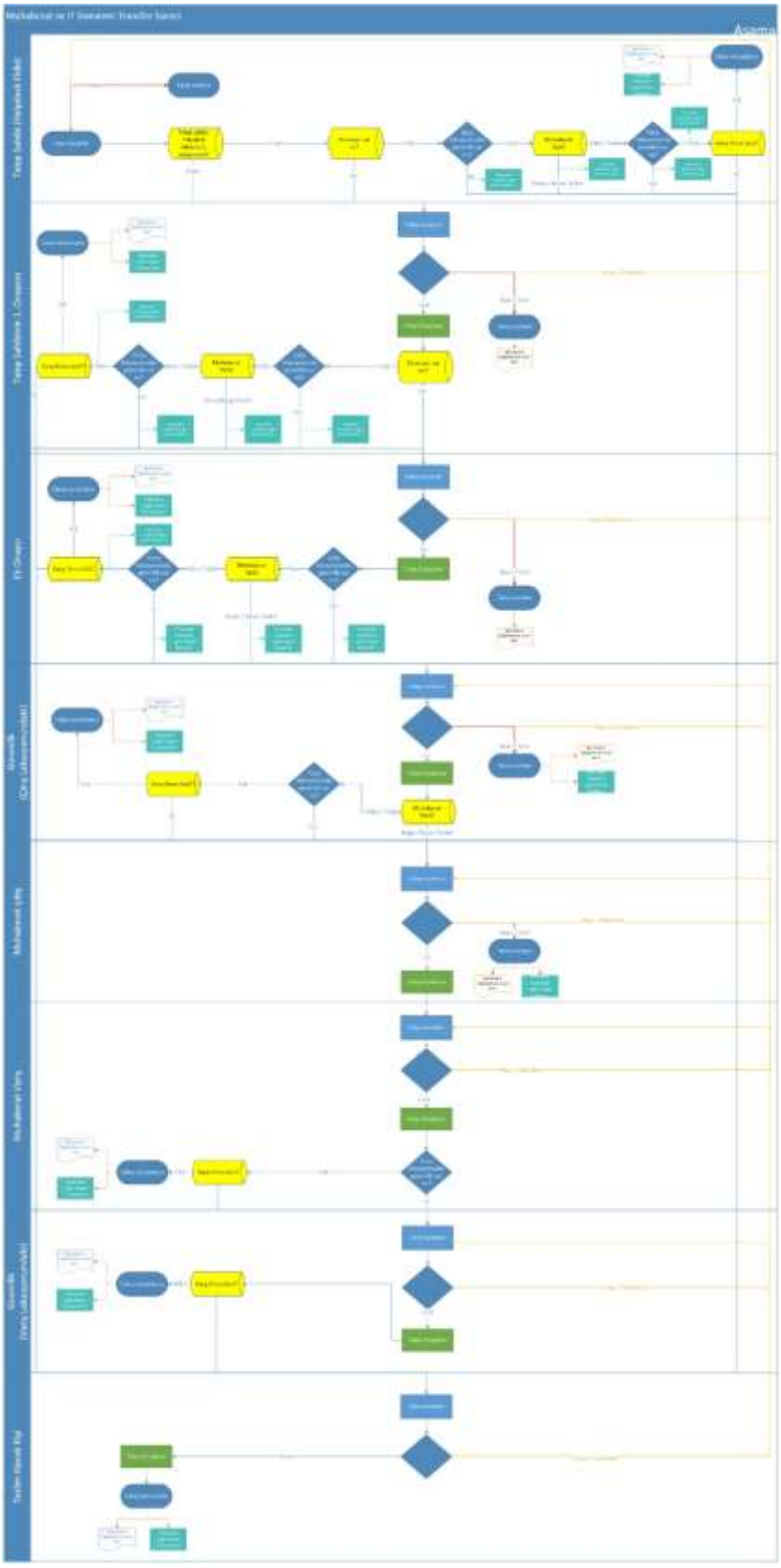
7 Birim ile görüşmeler sonucunda 5 farklı ürün ortaya çıktı. Bu ürünlerin dokümanları incelenip ilgili birimlerle tekrardan görüşmeler sağlandı ve 23 yeni özellik talebi ortaya çıktı. Bu özellikler hakkında teknik açıdan değerlendirilmesi için ilgili yazılım ekipleri ile bir araya gelindi.

3.3. Örnekleme

Yeni özellikler havuzundan yazılım testi için birkaç kritere göre özellik seçildi. Bu kriterler, yazılım ekiplerinin ilettiği takvime, tahmini efor süresine, etkilenen süreçlerin kapsam büyüklüğüne, entegre sistem sayısına, ilgili süreçlerin mevcut dokümanların yeterliliğine, test edilirken ekstra desteğe ihtiyaç duyma durumuna göre ilgili özellik belirlenmiştir.

3.4. Veriler ve Toplanması

Yazılım testi için belirlenen özellik ile ilgili süreç akışı talep sahipleri tarafından aktarıldı. Süreç akışı, akışı başlatan kişinin rolüne göre değişkenlik göstermekte olup, sistemde 4 farklı rol mevcuttur. Farklı akışların hepsi yazılım testi için birer senaryo demektir. Akış başlangıcından sonra ilerleme esnasında iken de farklı durumlara göre farklı sonuçlar alındığından bunlarda senaryoların sayılarını arttırmaktadır. Test edilecek toplam senaryoları görsel hale getirilmesi adına Visio programı kullanıldı. Şekil 3.1'deki Visio çiziminden anlaşıldığı üzere test edilmesi gereken senaryo sayısı da oldukça fazla olduğu gözlemlenmektedir.



Şekil 3.1 Seçili Özelliğin Süreç Akışı

3.5. Verilerin Yorumlanması

Yazılım testi yapılırken kullanılacak senaryolar manuel test olarak ilerletildi ve test dokümanı hazırlandı. Test dokümanı için bir şablon belirlendi. Test sonuçlarında sistemin beklenildiği gibi davranmadığı senaryolar tespit edilip yazılım ekibine bu senaryolar iletildi. Yazılım ekibi geliştirmeleri tamamladığında başarısız sonuçlanan senaryolar için tekrar testler yapıldı. Sonuç olarak, tüm senaryoların testten geçip son kullanıcıya sunulabilir denilmesi için geçen süre 1 ay sürdü. Bunun sadece 15 günü yazılım geliştirme diğer günleri ise test senaryoları ve UAT kabul süreci oldu. Aynı sistem test otomasyon sistemi ile test edilerek denendi. Manuel test ile 15 gün süren senaryoların test otomasyonu ile yapılması 2 gün sürdü. Buradaki ciddi maliyet kazancı ortaya çıkmıştır.

3.6. Süre ve Maliyet

Sonuç olarak, tüm senaryoların testten geçip son kullanıcıya sunulabilir denilmesi için geçen süre 1 ay sürdü. Bunun sadece 15 günü yazılım geliştirme diğer günleri ise test senaryoları ve UAT kabul süreci oldu. Aynı sistem test otomasyon sistemi ile test edilerek denendi. Manuel test ile 15 gün süren senaryoların test otomasyonu ile yapılması 2 gün sürdü. Buradaki ciddi maliyet kazancı ortaya çıkmıştır.

BÖLÜM 4. BULGULAR VE YORUMLAR

Bu bölümde, önce araştırma sonucunda elde edilen manuel test ve otomasyon testinin yapılabilmesi ilgili bulgular; sonra bulguları değerlendirerek anlamlandırmaya çalışıldığı yorumlara yer verilmiştir.

4.1. Bulgular

Projelerde Manuel Teste İhtiyaç Duyulma Sebepleri

- Otomasyon testlerin geliştirilen yazılım dışında kullanıma dayalı düşünülmeyen ve yazılımı yapılmamış senaryo için hata bulunması çok mümkün değildir.
- Kullanıcıların kullanım şekilleri ve alışkanlıklarına göre yaşanabilecek sorunları manuel test ile çok daha rahat bulunabilmektedir.
- Yazılım test uzmanından alınacak geribildirimler uygulamayı geliştiren yazılım uzmanı için değerlidir.
- Kullanıcı arayüzü işlevselliği sadece manuel test ile yapılabilir. Bu testler otomasyon ile pek mümkün olmaz.
- Projede değişen senaryolar (caseler) manuel test ile hızlı bir şekilde koşulabilir.
- Genelde birçok hata başka bir şeyin testini yaparken karşımıza çıkar.
- Sektörüne göre kimi testler fiziksel olarak saha da yapılması istenebilir.

Test Otomasyonu Gereksinimleri

Her süreci otomasyona çevirmek mümkün değildir. Bazı süreçler tahmin edildiğinden daha fazla efor gerektiren külfetli veya pahalı olabilir. Bir sürecin otomasyona uygun olup olmadığını anlamak için ön kontrolden geçirilmesi doğru olacaktır. Bu ön kontrol de aşağıdaki kriterlere yer verilebilir.

Tekrarlanabilme: İlgili sürecin tekrar kullanılabiliyor olması ve süreç içersinde duruma göre çoğaltılıyor olabiliyor ise test otomasyonuna çevrilmesindeki bu kriter için

uygundur. Ancak ilgili süreç sadece bir defa çalışıyor ve ürün içerisinde ileri aşamalarda karşımıza çıkmıyor ise bu sürece ayrılacak efor için tekrar düşünülmelidir.

Belirleyici: İlgili sürecin girdilerine göre çıktı sonucu belirli ise bu kritere uygundur. Aynı girdiler ile çıktı değişiyor ise test otomasyonuna dönüştürülmesi konusunda süreç akışı değiştirilebilir veya otomasyona dönüştürülmesi tekrar düşünülmelidir.

Ölçülebilme: Test otomasyon sistemleri kullanıcı kimliğine göre veya istisnai durumu gözetmeksizin sonucu tarafsız şekilde ölçebiliyor olması gerekmektedir.

4.2. Yorumlar

Literatür araştırması ve bulguları değerlendirdiğimizde yazılım dünyasın yazıl test uzmanlarının önemini ne kadar büyük olduğunu görmekteyiz. Sektörlerde ürünler git gide arttığından dolayı yazılım test uzmanlarına olan ihtiyaç artmaktadır. Manuel veya otomasyon testi kullan şirketler her zaman olacaktır. Ancak manuel testin yerini zamanla otomasyon sistemlerinin alacağı aşikardır.

Yazılım ürünleri içerisindeki akışların tamamen otomasyona geçmesi zor olduğundan öncelik sırasına göre geçiş yapılmaktadır. Buradaki maliyet şirketlerin kaçınmayacağı bir geçiş sürecidir. Bu geçiş süreci, destek veren şirketler sayesinde oldukça az sancılı olarak geçecektir.

BÖLÜM 5. SONUÇ

Bu bölümde sonuçlara dair özet, yargı ve öneriler alt bölümleri yer almaktadır. Yazılım dünyasında en büyük problemlerden biri geliştirilen güvenli ve hızlı bir şekilde test edilmesidir. Manuel ve otomasyon test türleri sektörde kullanılmaktadır. Manuel test zamanla yerini otomasyon testine bırakmaktadır. Otomasyon testi ile birlikte zaman, efor ve müşteri memnuniyeti konusunda kazançlar elde edilmektedir. Yazılım ürünlerinde her zaman geliştirme devam etmektedir. Bu geliştirmeleri talep eden kişiler en kısa sürede ilgili özelliği kullanmak isterler. İlgili özelliğin test edilmeden son kullanıcıya sunulması yanlıştır. Ancak bu test sürecinin uzun sürmesi de istenmemektedir. Bu durumda bu süreci kısaltacak ve hata riskini azaltacak test otomasyon yazılımları tercih edilmelidir. Buradaki kazanç gözle görülür düzeyde ve ciddi manada yüksektir. Manuel testler tamamen yok olmayacak ancak mümkün mertebe kullanılmaktan kaçınılmalıdır. Bazı testlerde en verimli test türü manuel test olduğu bilinilmektedir. Bu durumda manuel test her zaman hayatımızda olacaktır. Yazılım testinde kalitenin artması konusunda bulgularda ön kontrol denilen aşamadaki kriterler daha detaylandırılabilir ve arttırılabilir. Bu sayede ilgili sürecin test otomasyonuna geçişi konusunda karar daha tutarlı olacaktır.

Manuel testlerin kullanımın asla son bulmayacağını bulgulardan anlaşılmaktadır. Bu konuda fiziksel operasyon yürüten veya benzer şekilde süreçleri olan yazılımların test senaryoları sunulabilir ve mecburiyet daha net ortaya sunulabilir.

KAYNAKÇA

[1] Garousi, V., Coskunçay, A., Can, A. B., & Demirörs, O. (2013, September). Türkiye'deki Yazılım Test Uygulamaları Anketi. In UYMS.

[2] Yalçiner, B., Çiçek, S. S., Sahin, E., & Garousi, V. (2016, October). Yazilim güvenlik testi: Bir sistematik literatür Haritalamasi. In 10th Turkish National Software Engineering Symposium, UYMS 2016 (pp. 141-151).

[3] Sari, M., Özbek, M., & Kurt, A. (2015). Türkiye'deki Kamu Kurumları Yazılım Test ve Kalite Faaliyetleri Anketi, Değerlendirilmesi ve Öneriler. In UYMS.

[4] Demir, D., & KG, S. Endüstride yazılım testi ve kalite güvencesi etkinlikleri. I. Ulusal Yazılım Mühendisliği Sempozyumu, İzmir, 23, 25.

[5] Wilander, J., & Kamkar, M. (2002). A comparison of publicly available tools for static intrusion prevention. In 7th Nordic Workshop on Secure IT Systems," Towards Secure and Privacy-Enhanced Systems", 7-8 November 2002, Karlstad University, Sweden (p. 68). Karlstad University Studies.

[6] Thompson, H. H.(2003). Why security testing is hard.IEEE Security&Privacy,1, 83-86.

[7] Takanen, A., Vuorijärvi, P., Laakso, M., & Röning, J. (2004). Agents of responsibility in software vulnerability processes. Ethics and Information Technology, 6(2), 93-110.

[8] Mead, N. R., & Mcgraw, G. (2005). A portal for software security. IEEE Security & Privacy, 3(4), 75-79.

[9] Stytz, M. R., & Banks, S. B. (2006). Dynamic software security testing. IEEE security & privacy, 4(3), 77-79.

[10] Goertzel, K. M., Winograd, T., McKinley, H. L., Oh, L. J., Colon, M., McGibbon, T., ... & Vienneau, R. (2007). Software security assurance: a state-of-art report (sar). Informatıon assurance technology analysıs center (ıatac) herndon va.

[11] Zadeh, J., & DeVolder, D. (2007, March). Software development and related security issues. In Proceedings 2007 IEEE SoutheastCon (pp. 746-748). IEEE.

[12] Michael, C. C. (2005). Black box security testing tools.

[13] Epstein, J., Matsumoto, S., & McGraw, G. (2006). Software security and SOA: danger, Will Robinson!. IEEE Security & Privacy, 4(1), 80-83.

[14] Igure, V. M., & Williams, R. D. (2008). Taxonomies of attacks and vulnerabilities in computer systems. IEEE Communications Surveys & Tutorials, 10(1), 6-19.

[15] Godefroid, P., de Halleux, P., Nori, A. V., Rajamani, S. K., Schulte, W., Tillmann, N., & Levin, M. Y. (2008). Automating software testing using program analysis. IEEE software, 25(5), 30-37.

[16] Tøndel, I. A., Jaatun, M. G., & Jensen, J. (2008, April). Learning from software security testing. In 2008 IEEE International Conference on Software Testing Verification and Validation Workshop (pp. 286-294). IEEE.

[17] Shahriar, H., & Zulkernine, M. (2009, July). Automatic testing of program security vulnerabilities. In 2009 33rd Annual IEEE International Computer Software and Applications Conference (Vol. 2, pp. 550-555). IEEE.

[18] Bau, J., Bursztein, E., Gupta, D., & Mitchell, J. (2010, May). State of the art: Automated black-box web application vulnerability testing. In 2010 IEEE symposium on security and privacy (pp. 332-345). IEEE.

[19] Felderer, M., Agreiter, B., Zech, P., & Breu, R. (2011). A classification for model-based security testing. Advances in System Testing and Validation Lifecycle (VALID 2011), 109-114.

[20] Liu, B., Shi, L., Cai, Z., & Li, M. (2012, November). Software vulnerability discovery techniques: A survey. In 2012 fourth international conference on multimedia information networking and security (pp. 152-156). IEEE.

[21] Ami, P., & Hasan, A.M. (2012). Seven Phrase Penetration Testing Model. International Journal of Computer Applications, 59, 16-20.

[22] Kates, J. (2001). Security Testing Is Not All the Same: A Reference Taxonomy. In Architectures for E-Business Systems (pp. 607-616). Auerbach Publications.

İnternet Kaynakları:

[URL-1]https://www.ekonomist.com.tr/teknoloji/yazilim-testi-sektoru-buyume-ataginda.html

Erişim Tarihi: 02.06.2014

[URL-2] https://www.innova.com.tr/tr/blog/buyuk-veri-blog/test-otomasyonu-ile-yazilim-testleri-nasil-kolaylasiyor

Erişim Tarihi: 02.06.2014

[URL-3] https://medium.com/n%C3%B6bet%C3%A7i-test%C3%A7i/yaz%C4%B1l%C4%B1m-test-s%C3%BCre%C3%A7leri-73eba7d04694
Erişim Tarihi: 02.06.2014
[URL-4] https://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/08/yaz%C4%B1l%C4%B1m-testi-ve-test-s%C3%BCre%C3%A7leri
Erişim Tarihi: 02.06.2014
[URL-5] https://medium.com/kuka-mobile-tech/manuel-test-nedir-aefc4757983a
Erişim Tarihi: 02.06.2014
[URL-6] https://tr.myservername.com/manual-testing-vs-automation-testing-what-is-difference
Erişim Tarihi: 02.06.2014