Web Tabanlı Uygulama Üzerine Otomasyon Yazılım Testi

Oğuzhan İnce
oguzhan.ince@ogr.sakarya.edu.tr

SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

Özet — Çevik test, bir yazılım test uygulamasıdır, çevik politika kurallarını izler ve test sürecindeki bir müşteri gibi kritik bir bölüm olarak yazılım geliştirmeyi dikkate alır. Gerekli insan gücü miktarını en aza indirmek için bunu yapmak için otomatik testler kullanılır. Bu yazıda geleneksel bir otomasyon test modeli tartışılmıştır. Otomatik çevik testler için bir model önerilmiş ve deneysel bir çalışma da bir Web uygulamasının testinde gösterilmiştir[1]. Sonuçlar çevik test modeli kullanılarak değerlendirilir ve geleneksel ve çevik test modelleri arasında bir karşılaştırma yapılır.

Anahtar Kelimeler — Agile geliştirme, çevik geliştirme, yazılım testi, otomasyon testi, web tabanlı uygulama testi, selenium aracı

I. GIRIŞ

Bir yazılım test projesi, önemli bir tahmin ve yazılım geliştirme yaşam döngüsünde tam bir yürütme ile başarılı olabilir [2]. Yazılım testi, sistem davranışına ve müşteri ihtiyacına göre birim, entegrasyon, sistem ve kabul testi gibi farklı düzeylerde testlerin yapıldığı yazılım geliştirmenin ana parçasıdır; birim ve entegrasyon testi ise bireysel modüllere odaklanır. sistem ve kabul testi sistemin genel davranışına odaklanır [3]. Teknoloji ve yazılım geliştiriciler tarafından en son teknoloji eğilimlerinin ve gelişim uygulamalarının benimsenmesi, teknolojilerin ve yazılım geliştirme uygulamalarının güçlü ve zayıf yönleriyle ilgili fikirlerin kutuplaşmasına yol açmaktadır [4]. Otomasyon formları çevik testlerin en önemli parçalarından biridir; Aksi takdirde, çevik kalkınma programına ayak uydurmak çok zordur.

Bu aşamada otomasyonla tespit edilen hataların önceliklendirilmesi ve sprint sırasındaki sabitlenmesi süreci de belirlenir. Regresyon testi durumunda, test vakalarının tekrar tekrar yazılımda uygulandığı otomasyon testi ile verimlilik artar. Çevik testlerde, genellikle kod yineleme sırasında yazılır ve test işlemi her yinelemeden sonra yapılır [5]. Geliştiriciler ve test edenler, hata oluşma olasılığını azaltmak için birlikte çalışırlar. Geliştiriciler birim testini yapar ve geri kalan sistem testi de kabul testi olarak da bilinen müşteri tarafından yapılır ve bu istemcilere çevik olarak mükemmellik garantisi hareketine genel bir bakış veren geri bildirimlerini sağlar [6].

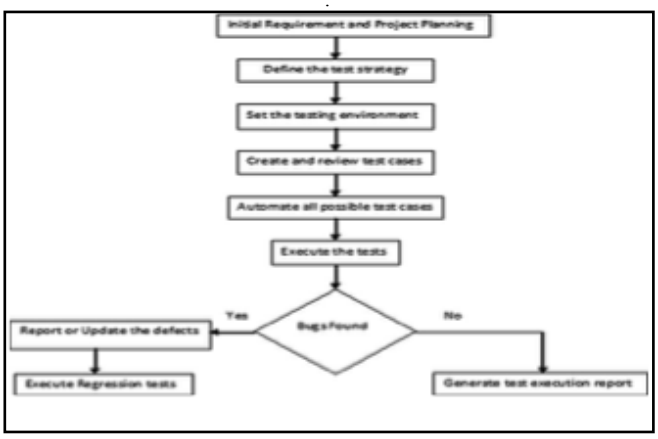
II. LITERATÜR İNCELEMESI

Anket sonuçlarında, test vakalarının sadece %26'sının otomasyon testi ile yapıldığı ve bunun geçen yıllara göre oldukça düşük olduğu ortaya kondu. Kağıt, otomasyon testi ve araçları konusunda daha fazla çaba göstermeye odaklanmıştır [7]. Bir anket raporuna göre, "Çevik kalkınma durumu", katılımcıların % 80'inden fazlasının bir düzeyde çevik test metodolojisi benimsediğini ve katılımcıların diğer yarısının organizasyonlarının yaklaşık iki ila üç arasında çevik test metodolojisi kullandığını göstermiştir [8]. Çevik yazılım geliştirme yöntemleri bugünlerde yaygın olarak genişletilmekte ve kurulmaktadır. Çevik test metodolojisini benimsemenin temel nedeni, ürünü bir süre içinde teslim etmek, verimliliği arttırmak ve sık sık değişen iş gereksinimlerini kolayca yönetmektir [9]. Bir anket raporu çevik testlerin çoğunlukla geliştirme süreci olduğunu belirtti. Çevik testler küçük yinelemeler (% 79), düzenli geri bildirim (% 77) ve günlük bazda (% 71) yapılan scrum toplantısı en önemli faktörlerdir [10]. Hanssen Geir Kjetil, çevik test metodolojisinin kullanımının küresel olduğunu belirtti [11].

Değerlendirme ve zamanlama, herhangi bir boyut ve sonuçtaki bir projenin yazılım büyümesinin gerçekleştirilmesinin ana kaygısıdır; Bir projenin çevik yazılım büyümesi, mahkumiyetle ilgili tartışmaların yapıldığı alan olmuştur [1].

III. OTOMATIK TEST

Şekil 1'de, farklı adımların belirtildiği otomasyon test modelinden bahsedilmiştir. Uygulama sürecinden sonra, test ortamı ayarlanır, tüm test faaliyetlerini planlayın. Tüm planlamalarda, kıdemli test uzmanları tarafından gözden geçirilen farklı test senaryoları oluşturulur [12]. Test vakaları yürütülür ve hata raporu oluşturulur. Eğer buglar üretilmezse, yinelemeler tamamlanır ve biriktirme hikayeleri sona erer [13].



Şekil 1. Geleneksel ortamda otomatikleştirilmiş test modeli.

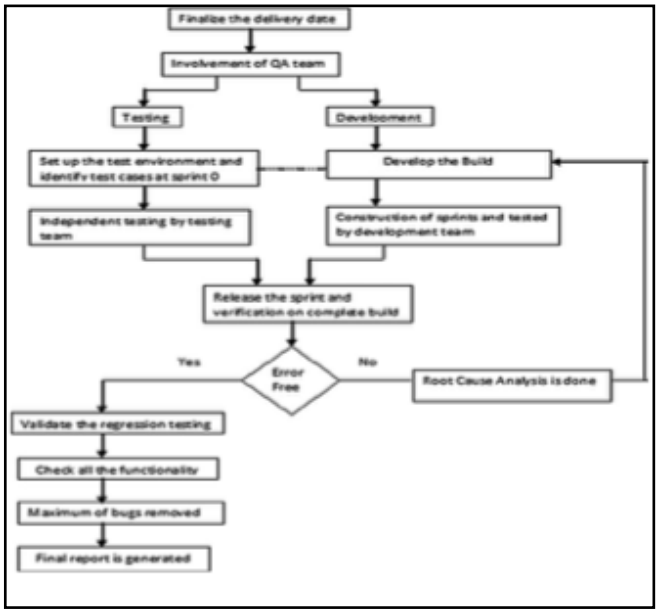
IV. ÖNERILEN YÖNTEM

Çevik ortamda, yazılımın doğruluğunu artırmak için projenin denenmesi önerilen bir yöntemle gerçekleştirilir. Çevik bir gelişmede, geliştiriciler de testten sorumludur. Bu nedenle, geliştiriciler ve test edenler arasında mümkün olan paralel ve bağımsız testler vardır. Şekil 2, çevik metodoloji kullanan ve önerilen otomasyon test modelidir [1].

Aşağıdaki algoritma, test paketlerini geliştiriciler ve test edenler tarafından yürütmek için otomatik çevik testlerin sözde kodudur. 2. satırda, başlangıçta projeye dahil olan

kalite güvence ekibi. Satır 3, paralel testin geliştiriciler ve test edenler tarafından yapıldığını ve yazılımın işlevselliğini göstermek için stand toplantıları yapıldığını gösterir.
4. satırda, test ediciler farklı sprint çevrimlerinde otomasyon testlerini oluşturuyor ve 5. satırda doğrulama testi oluşturuluyor. Test paketi sonucu başarısız olursa 6. satırda 7. satırda, hataları gidermek için RCA (kök neden analizi) gerçekleştirilir. 6. satırda, sonuç başarıyla geçtiğinde, 10. satırdaki regresyon testini doğrulayın ve testi çalıştırın ve sonlandırın. Son olarak 13. satırda rapor oluşturulur [1].

1. Dd <— Set the delivery date
2. QA <—> P Involvement of QA team in project
3. Deploy the build in the Pre-Production environment.
4. t <—> d Testing and development both are implemented parallel.
5. T —> Generate automation test suite, in the sprint cycle.
6. Release the 1st sprint and complete the BVT
7. **If** d's and t's result failed **then**



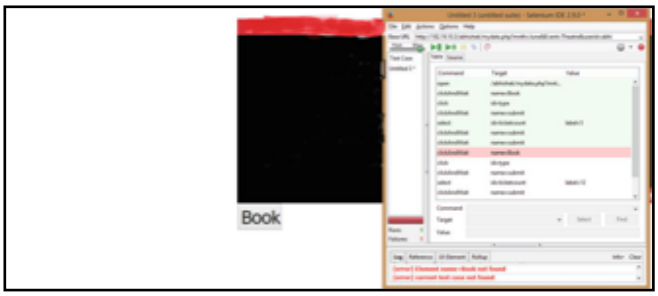
Şekil 2. Çevik metodoloji kullanılan yazılım test modeli

8. **do** RCA ("remove the bugs in next sprint")
9. GOTO step 3
10. **else**
11. T —> RT (validate the regression testing)
    Result <— T. runtest
12. **end if**
13. Generate the report

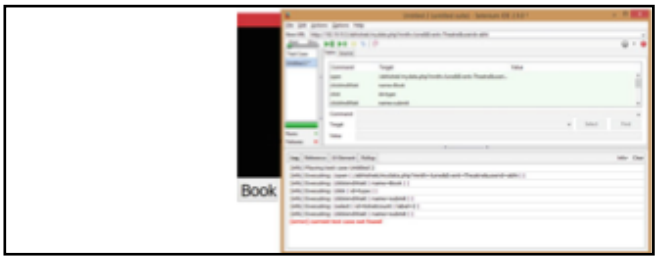
Whereas; Dd: Delivery Date; QA: Quality Assurance Team; P: Project; T: Tester; D: Development; BVT: Build Verification Test; t: testing; d: development; RCA: root cause analysis; RT: regression testing.

A. Deneysel Çalışma

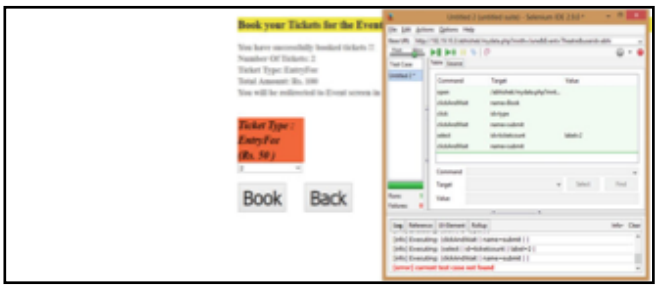
Web uygulamasının uygulanması sırasında, çevik metodoloji kullanılarak proje geliştirilmiş ve test edilmiştir. Hem ürün geliştirme hem de otomasyon testleri titizlikle uygulandı. Çevik ortamda uygulama sırasında geliştiriciler ve test ekipleri paralel çalıştı. Selenium aracı kullanılarak farklı adımlarda otomasyon test uygulaması yapılmıştır. Çevik test modellerinin adımlarını kullanarak, ilk testte, Şekil 3'te gösterildiği gibi hata bulundu. RCA yapıldı ve daha sonra yeni bir sprint döngüsü uygulandı. Hata, Şekil 4'te gösterildiği gibi çözüldü ve regresyon testini doğruladı. Nihai sonuç, Şekil 5'teki gibi değerlendirildi ve maksimum hata giderildi ve bir nihai rapor üretildi [1].



Şekil 3. İlk testte hataların tespiti



Şekil 4. Selenium IDE kullanarak book butonunu test etme



Şekil 5. Bilet rezervasyonu başarıyla yapıldı.

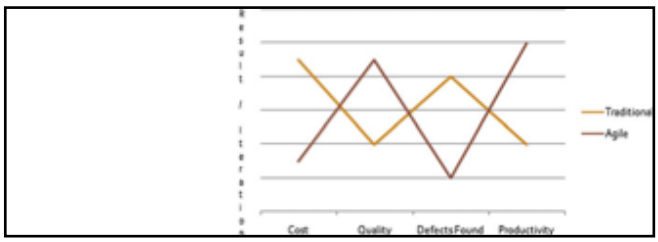
B. Geleneksel ve Çevik Otomatikleştirilmiş Test Modelinin Karşılaştırılması ve Sonucu

Web uygulaması tersi geleneksel ve çevik test modelleri kullanılarak test edilmiştir. Çevik test modellerinde, sonuçlar geleneksel modelden daha iyi hale getirildi. Çevik bir metodoloji içeren veya içermeyen test uygulaması sırasında sonuçlar Tablo a'daki gibi gözlendi:

Çevik ve geleneksel ortamlardaki testler arasındaki karşılaştırma, Şekil 6'da gösterilmektedir. Şekildeki sonuçlar, yineleme başına farklı parametrelere dayanmaktadır. Parametreler maliyet, kalite, tespit edilen hatalar ve verimliliktir [1].

Çevik Test Modeli	Otomatik Test Modeli
Projenin teslim tarihine göre, geliştirme sürecine paralel olarak testler yapıldı.	Test, geliştirme sürecinin tamamlanmasından sonra gerçekleştirildi.
Geliştirmenin ilk aşamasından itibaren yapılan paralel testler nedeniyle, sonunda daha az sayıda kusur değerlendirildi.	Test müşterinin katılımı olmadan yapıldı ve müşteri memnuniyeti için daha fazla zaman aldı.
Regresyon testi tek bir sprint içinde sık sık yapıldığından zaman ve maliyetten tasarruf sağladı	Geliştirme sürecinin tamamlanmasından sonra, test senaryoları ve ardından otomatik test senaryoları oluşturuldu
Müşterilerin sıralı katılımı ile üretkenlik ve ürün kalitesi iyileştirildi	Düşük ürün kalitesiyle daha az üretkenlik

a. Çevik model ile otomatik test modelinin karşılaştırılması



Şekil 6. Çevik uygulama ve geleneksel uygulama kullanarak parametreleri test etme.

V. SONUÇ

Rapor, otomasyon testinin sadece test kapsamını arttırmakla kalmayıp, aynı zamanda maliyeti düşürdüğü ve ürünün teslimatını iyileştirdiği sonucuna varmaktadır. Bu yazıda çevik ortam ve geleneksel gelişim kullanarak bir Web uygulamasında bir otomasyon testinin uygulandığı deneysel bir çalışma ele alınmıştır. Önerilen çevik test modeli, üretim ekibiyle planlı ve organize bir şekilde çalıştı. Sonunda, çevik ve geleneksel otomatik test modelleri arasında karşılaştırma vardır ve bu durum çevik test yoluyla elde edilen sonuçlar geleneksel testlerden daha iyidir [1].

REFERENCES

1. Saru Dhir, Deepak Kumar: Software Engineering, Advances in Intelligent Systems and Computing (AISC, volume 731) Automation Software Testing on Web-Based Application pp. 691–698 (2019)
2. Sharma, M.A., Kushwaha, D.S.: A metric suite for early estimation of software testing effort using requirement engineering document and its validation. In: Computer and Communication Technology (ICCCT) 2nd International Conference, pp. 373–378 (2012)
3. Spillner, A., Linz, T., Schaefer, H.: Software Testing Foundation: A Study Guide for the Certified Tester Exam, 3rd ed. Rocky Nook (2011)
4. Aggarwal, S., Dhir, S.: Ground axioms to achieve movables: methodology. Int. J. Comput. Appl. **69**(14) (2013)
5. Karhu, K., Repo, T., Taipale, O., Smolander, K.: Empirical observations on software testing automation, international conference on software testing verification and validation (ICST'09). IEEE Computer Society, Washington, DC, USA, pp. 201–209. DOI = 10.1109/ ICST.2009.16 (2009)
6. Aggarwal,S.,Dhir,S.:Swifttack:anewdevelopmentapproach.In:InternationalConference on Issues and Challenges in Intelligent Computing Techniques, IEEE (2014)
7. Kasurinen, J., Taipale, O., Smolander, K.: Software test automation in practice: empirical observations. Adv. Softw. Eng., Article 4 (2010)
8. http://www.versionone.com/pdf/ 2011_State_of_Agile_Development_Survey_Results.pdf
9. Rai, P., Dhir, S.: Impact of different methodologies in software development process. Int. J. Comput. Sci. Inf. Technol. 5(2), 1112–1116 (2014). ISSN: 0975-9646
10. West, D., Grant, T.: Agile development: mainstream adoption has changed agility. Forrester Research Inc. (2010)
11. Hanssen, G.K., Šmite, D., Moe, N.B.: Signs of agile trends in global software engineering research: a tertiary study. In: 6th International Conference on Global Software Engineering, pp. 17–23 (2011)
12. Collins.E.,Macedo.G.,Maia.N.,Neto.A.D.:Anindustrialexperienceonthe applicationof distributed testing in an agile software development environment. In: Seventh International Conference on Global Software Engineering, IEEE, pp. 190–194 (2012)
13. Mattsson, A., Lundell, B., Lings, B., Fitzgerald, B.: Linking modeldriven development and software architecture: A case study. IEEE Trans. Softw. Eng. **35**(1), 83–93 (2009)
14. Geir Kjetil Hanssen, Tor Stålhane, Thor Myklebust: SafeScrum® – Agile Development of Safety-Critical Software (2018)