T.C.
DOKUZ EYLÜL ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
YÖNETİM BİLİŞİM SİSTEMLERİ ANABİLİM DALI
YÖNETİM BİLİŞİM SİSTEMLERİ PROGRAMI
YÜKSEK LİSANS TEZİ

WEB UYGULAMALARINDA TEST OTOMASYONU: ARAÇLARIN KARŞILAŞTIRILMALI ANALİZİ

Emre PALA

Danışman
Dr.Öğr. Üyesi Hakan AŞAN

2025

T.C.
DOKUZ EYLÜL ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
YÖNETİM BİLİŞİM SİSTEMLERİ ANABİLİM DALI
YÖNETİM BİLİŞİM SİSTEMLERİ PROGRAMI
YÜKSEK LİSANS TEZİ

WEB UYGULAMALARINDA TEST OTOMASYONU: ARAÇLARIN KARŞILAŞTIRILMALI ANALİZİ

Emre PALA

Danışman
Dr.Öğr. Üyesi Hakan AŞAN

İZMİR - 2025

YÜKSEK LİSANS
TEZ ONAY SAYFASI

Üniversite	: Dokuz Eylül Üniversitesi
Enstitü	: Sosyal Bilimler Enstitüsü
Adı ve Soyadı	: EMRE PALA
Öğrenci No	: 2022800662
Tez Başlığı	: Web Uygulamalarında Test Otomasyonu : Araçların Karşılaştırmalı Analizi
Savunma Tarihi	: 31.07.2025
Danışmanı	: Dr.Öğr.Üyesi Hakan AŞAN

JÜRİ ÜYELERİ

Ünvanı, Adı, Soyadı	**Üniversitesi**	**İmza**
Dr.Öğr.Üyesi Hakan AŞAN	- Dokuz Eylül Üniversitesi	.....
Prof. Dr. Vahap TECİM	- Dokuz Eylül Üniversitesi	.....
Doç. Dr. Ejder AYÇİN	-Kocaeli Üniversitesi	.....

EMRE PALA tarafından hazırlanmış ve sunulmuş olan bu tez savunmada başarılı bulunarak oy birliği( ) / oy çokluğu( ) ile kabul edilmiştir.

Prof. Dr. Erhan DEMİRELİ
Müdür

YEMİN METNİ

Yüksek lisans tezi olarak sunduğum “Web Uygulamalarında Test Otomasyonu: Araçların Karşılaştırılmalı Analizi” adlı çalışmanın, tarafımdan, akademik kurallara ve etik değerlere uygun olarak yazıldığını ve yararlandığım eserlerin kaynakçada gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve bunu onurumla doğrularım.

16.6.2025

Emre PALA

ÖZET

Yüksek Lisans Tezi

Web Uygulamalarında Test Otomasyonu: Araçların Karşılaştırılmalı Analizi

Emre PALA

Dokuz Eylül Üniversitesi

Sosyal Bilimler Enstitüsü

Yönetim Bilişim Sistemleri Anabilim Dalı

Yönetim Bilişim Sistemleri Programı

Yıllar içerisinde yazılım teknolojilerinin gelişimine paralel olarak web uygulamaları, bireylerin günlük yaşamına ve iş süreçlerine gün geçtikçe daha fazla entegre olmuş ve birçok alanda kritik bir rol üstlenmiştir. Web uygulamalarının sürekli gelişimi ve değişimi, uygulamaların karmaşıklığını artırmakta ve bunun sonucu olarak da yazılımların kontrol edilmesi zorlaşmaktadır. Bu bağlamda, web uygulamaları kullanan firmaların ve son kullanıcıların karşılaşabileceği olası zararları önlemek amacıyla yazılım test otomasyonu kavramı önem kazanmıştır. Yazılım test otomasyonu, yazılımın kalitesi ve güvenilirliğinin kontrol edilmesinde önemli bir araçtır ve zaman geçtikçe karmaşıklaşan web uygulamaları için yazılım geliştirme süreci kadar kritik bir süreç olarak değerlendirilmektedir. Test otomasyonu süreçlerinin başarısı büyük ölçüde otomasyon aracının doğru ve etkin biçimde seçilmesine bağlıdır. Benzer şekilde test aracının yanlış seçimi de test sürecinin etkinliğini ve kalitesini doğrudan olumsuz etkilemektedir. Test aracının seçimine yönelik pratikte ve literatürde birçok çalışma bulunmaktadır. Literatürde yer alan çalışmalar, genel olarak yazılım test araçlarını genel amaçlar çerçevesinde karşılaştırmış olup, spesifik olarak web uygulamalarına yönelik araç seçimi konusunda yeterince odaklanmamıştır. Bu çalışmanın özgünlüğü ise doğrudan web uygulamaları için geliştirilmiş ve popüler olan test otomasyon araçlarını detaylı biçimde inceleyip karşılaştırmasında bilimsel bir yöntem olan çok kriterli karar verme yöntemlerinden yararlanmasındadır. Test otomasyonu seçimi için kullanılacak veriler alanında uzman kişilerle gerçekleştirilen görüşmeler

neticesinde toplanmıştır. Görüşmelerden elde edilen veriler öncelikle Geliştirilmiş Bulanık SWARA yöntemi ile kriterler ağırlaklandırılmış. Stability (Kararlılık) kriteri, en önemli araç seçim kriteri olarak belirlenmiş; bunu, sırasıyla Doküman ve Topluluk Desteği kriteri ile Güvenlik kriteri, ikinci ve üçüncü en önemli kriterler olarak izlemiştir. Öte yandan, Raporlama Kalitesi kriteri ise uzman görüşlerine göre en az önemli kriter olarak tespit edilmiştir. Daha sonra ağırlıkları belirlenen kriterlere göre Bulanık MARCOS yöntemi kullanılarak araçlar analiz edilmiştir. Yapılan analizlerin sonucunda, incelenen araçlar arasında Selenium aracının, web uygulamaları otomasyon test süreçleri için diğer alternatiflere kıyasla daha uygun ve etkin bir çözüm sunduğu belirlenmiştir fakat Cypress aracı ile aralarında performans olarak minimal farklar olması Selenium aracı ile Cypress aracı arasında çok minimal bir performans farkı olduğunu ortaya koymaktadır. Yapılan çalışma, pratik açıdan karar vericilere yol gösterici nitelik taşırken, aynı zamanda yazılım testi alanında literatüre önemli bir katkı sunmaktadır.

Anahtar Kelimeler: Yazılım Testi, Web Uygulamaları, Test Otomasyon, Çok Kriterli Karar Verme, Geliştirilmiş Bulanık SWARA, Bulanık MARCOS

ABSTRACT

Master's Thesis

Test Automation in Web Applications: Comparative Analysis of Tools

Emre PALA

Dokuz Eylül University

Graduate School of Social Sciences

Department of Management Information Systems

Management Information Systems Program

Over the years, in parallel with the development of software technologies, web applications have become increasingly integrated into individuals' daily lives and business processes, playing a critical role in many areas. The continuous evolution and transformation of web applications have led to increased complexity, thereby making software management and control more challenging. In this context, software test automation has gained significant importance to prevent potential damages faced by companies and individual users relying on web applications. Software test automation is a vital tool for ensuring software quality and reliability and is considered as critical as software development, particularly for increasingly complex web applications. The success of test automation processes largely depends on selecting the appropriate and effective automation tool. Incorrect tool selection directly and adversely affects the efficiency and quality of the test processes. Many previous studies in the literature have compared various software testing tools generally, without adequately focusing on tools specifically designed for web applications. The originality of this study lies in its detailed examination and comparison of popular test automation tools developed specifically for web applications. Ethical approval required for this research was obtained from the Dokuz Eylül University Social and Humanities Sciences Research and Publication Ethics Committee with decision number 19, dated April 22, 2025. Data collected through interviews conducted with experts in test automation were analyzed using the Improved Fuzzy SWARA and Fuzzy MARCOS methods. The Stability criterion was identified as the most

important tool selection criterion, followed by the Documentation and Community Support criterion and the Security criterion as the second and third most important criteria, respectively. On the other hand, the Reporting Quality criterion was determined, based on expert opinions, to be the least important criterion. Subsequently, the tools were analyzed using the Fuzzy MARCOS method according to the criteria whose weights had been determined. The results of the analyses indicated that, among the tools examined, Selenium provides a more suitable and effective solution for web application automation testing processes compared to other alternatives. However, the presence of only minimal performance differences between Selenium and Cypress reveals that the performance gap between these two tools is negligible. The study serves as a practical guide for decision-makers while also making a significant contribution to the literature in the field of software testing.

Keywords: Software Test, Web Applications, Test Automation, Multi-Criteria Decision Making, Improved Fuzzy SWARA, Fuzzy MARCOS

WEB UYGULAMALARINDA TEST OTOMASYONU: ARAÇLARIN KARŞILAŞTIRILMALI ANALİZİ

İÇİNDEKİLER

TEZ ONAY SAYFASI ii
YEMİN METNİ iii
ÖZET iv
ABSTRACT vi
İÇİNDEKİLER viii
KISALTMALAR x
TABLOLAR LİSTESİ xi
ŞEKİLLER LİSTESİ xiii

GİRİŞ 1

BİRİNCİ BÖLÜM
LİTERATÜR ARAŞTIRMASI

1.1. LİTERATÜR ARAŞTIRMASI 6
1.2 LİTERATÜR ÖZET 14

İKİNCİ BÖLÜM
WEB UYGULAMALARI VE TEST OTOMASYON

2.1. WEB UYGULAMASI 16
2.1.1. Web Uygulamalarının Tarihsel Süreci 17
2.2. YAZILIM TESTİ 21
2.2.1. Test Otomasyon 23
2.2.2. Test Otomasyon Araçları 24
2.2.2.1. Selenium 26
2.2.2.1.1. Selenium Özellikleri 27
2.2.2.1.2. Selenium Kullanım Alanları 28
2.2.2.2. Cypress 28

2.2.2.2.3. UI Coverage ve Cypress Accessbility Özellikleri 31
2.2.2.2.4. Cypress Kullanım Alanları 32
2.2.2.3. Puppeteer 33
2.2.2.3.1. Puppeteer Özellikleri 33
2.2.2.3.2. Puppeteer Kullanım Alanları 34
2.2.2.4. Playwright 35
2.2.2.4.1. Playwright Özellikleri 35

ÜÇÜNCÜ BÖLÜM
ÇOK KRİTERLİ KARAR VERME

3.1. ÇOK KRİTERLİ KARAR VERME TANIMI 37
3.2. BULANIK MANTIK VE ÇKKV İLİŞKİSİ 39
3.2.1. Bulanık Küme 40
3.3. SWARA METHODU 41
3.3.1. Geliştirilmiş Bulanık SWARA Yöntemi 42
3.4 MARCOS Methodu 46
3.4.1. Bulanık Marcos Yöntemi 46

DÖRDÜNCÜ BÖLÜM
UYGULAMA

4.1 GELİŞTİRİLMİŞ BULANIK SWARA ÇÖZÜMÜ 54
4.1.1. Uzmanların Değerlendirmeleri 55
4.1.2. Kriter Ağırlıklandırma Tablosu 56
4.2 BULANIK MARCOS ÇÖZÜMÜ 60

SONUÇ 66
KAYNAKÇA 68

KISALTMALAR

ÇKKV	Çok kriterli karar verme
WWW	Dünya Çapındaki Ağ
AI	Yapay Zeka
AR	Arttırılmış Gerçeklik
NPM	Node Paket Yönetimi
RC	Uzaktan Komtrol
IDE	Bütünleşik Geliştirme Ortamı
CI/CD	Continuous Integration/Continuous Delivery
SDLC	Yazılım Geliştirme Yaşam Döngüsü
ERP	Kurumsal Kaynak Planlama
SWARA	Adım adım ağırlık değerlendirme oranı analizi
MARCOS	Karmaşık Orantılı Değerlendirme
BNP	En iyi Bulanık Olmayan Performans

TABLOLAR LİSTESİ

Tablo 1: Literatür Araştırmalarına Dair Özet Tablo s. 14
Tablo 2: Web 1.0 ve Web 2.0 özellikleri s. 19
Tablo 3: Selenium Bileşenleri s. 26
Tablo 4: Selenium Özellikleri s. 27
Tablo 5: Selenium Kullanım Alanları s. 28
Tablo 6: Cypress Ürünleri s. 29
Tablo 7: Cypress Uygulaması Özellikleri s. 30
Tablo 8. Cypress Cloud Uygulamasının Özellikleri s. 30
Tablo 9: Ui Coverage ve Cypress Accessibility Ürünleri Özellikleri s. 31
Tablo 10. Cypress Kullanım Alanları s. 32
Tablo 11. Puppeteer Aracı Özellikleri s. 33
Tablo 12. Puppeteer Aracı Kullanım Alanları s. 34
Tablo 13: Playwright Aracının Özellikleri s. 35
Tablo 14: Playwright Aracı Kullanım Alanları s. 36
Tablo 15: Objektif Ve Sübjektif Yöntemler s. 39
Tablo 15: Dilsel Değişkenler Ölçeği s. 43
Tablo 16: SWARA ve Geliştirilmiş Bulanık SWARA İle Yapılan Çalışmalar s. 44
Tablo 17: MARCOS Dilsel Değişkenler Ölçeği s. 47
Tablo 18: MARCOS ve Bulanık MARCOS ile yapılmış çalışmalar s. 50
Tablo 19: Uzman Bilgileri s. 53
Tablo 20. MARCOS Bulanık Sayılar s. 54
Tablo 21. Kriterlerin İsmi s. 54
Tablo 22. Örnek Kriter Değerlendirmesi s. 55
Tablo 23. Birinci Uzmana Ait Kriter Değerlendirme Tablosu s. 55
Tablo 24: İkinci Uzmana Ait Kriter Değerlendirme Tablosu s. 56
Tablo 25: Birinci Uzmana Ait Bulanık Ağırlık Değerler Tablosu s. 57
Tablo 26: İkinci Uzmana Ait Bulanık Ağırlık Değerler Tablosu s. 57
Tablo 27: Birinci Uzmana Ait Durulaştırılan Ağırlık Değerler Tablosu s. 58
Tablo 28: İkinci Uzmana Ait Durulaştırılan Ağırlık Değerler Tablosu s. 58
Tablo 29: Normalize Edilen Nihai Ağırlık Tablosu s. 59

Tablo 30: Uzman 1 Alternatif Değerlendirmeleri s. 60
Tablo 31: Uzman 2 Alternatif Değerlendirmeleri s. 61
Tablo 32: Başlangıç Bulanık Karar Matrisi s. 61
Tablo 33: Bulanık Genişletilmiş Karar Matrisi s. 62
Tablo 34: Normalize Karar Matrisi s. 62
Tablo 35: Bulanık Ağırlık Değerleri s. 63
Tablo 36: Ağırlıklandırılmış Bulanık Karar Matrisi s. 63
Tablo 37: Nihai Bulanık Karar Matrisi s. 63
Tablo 38: Alternatiflerin İdeal Fayda Derecesi s. 64
Tablo 39: Alternatiflerin Anti-ideal Fayda Derecesi s. 64
Tablo 40: Ti Matrisi s. 64
Tablo 41: İdeal Fayda Fonksiyonu f(k+) s. 65
Tablo 42: Anti-ideal Fayda Fonksiyonu f(k-) s. 65
Tablo 43: Nihai Sıralama Tablosu s. 65

ŞEKİLLER LİSTESİ

Şekil 1: Kavramsal Çerçeve s. 5
Şekil 2. Yazılım Geliştirme Yaşam Döngüsü Adımları s. 22
Şekil 3: Araçların günlük indirme sayıları s. 25
Şekil 4: Selenium Mimarisi s. 27
Şekil 5: ÇKKV Süreci s. 38
Şekil 6: Üçgen Bulanık Sayı s. 41

GİRİŞ

Son yıllarda web tabanlı yazılımlar (web uygulamaları), dijital çağın bir gereği olarak bireylerin günlük yaşamının ve kurumların iş süreçlerinin ayrılmaz bir parçası haline gelmiştir. Bugün e-ticaret alışverişlerinden sosyal medya ve eğitim platformlarına kadar hemen her alanda web uygulamaları yaygın şekilde kullanılmaktadır (Cen ve Zhao, 2024: 1). Web teknolojilerinin ulaştığı ölçek de dikkat çekicidir: Örneğin 2023 yılı ilk çeyreği itibarıyla dünya genelinde kayıtlı alan adı sayısının 354 milyonu aştığı ve 2024 itibarıyla internet kullanıcı sayısının 5,44 milyara ulaştığı rapor edilmiştir (Kertusha vd., 2025a: 1). Elde edilen veriler, web tabanlı hizmet ve uygulamaların küresel ölçekte ne denli yaygınlaştığını ortaya koymaktadır.

Web uygulamalarının hayatın her alanında benimsenmesi, beraberinde büyük bir sistem karmaşıklığı ve yazılım yoğunluğu getirmektedir. Güncel web sistemleri, çoğu zaman kesintisiz hizmet vermesi beklenen, büyük ölçekli ve birbirine bağlı bileşenlerden oluşan yapılardır. Nitekim günümüz toplumları, sürekli erişilebilirlik gerektiren karmaşık ve büyük ölçekli yazılım yoğun sistemlere giderek daha bağımlı hale gelmiştir; mevcut durum modern web sistemlerin geliştirilmesini, bakımını ve testini oldukça güçleştirmektedir (Venters ve diğerleri, 2023). Başka bir deyişle, web tabanlı sistemler büyüdükçe ve daha fazla bileşen barındırdıkça, bu sistemlerde ortaya çıkabilecek güvenlik açıkları, performans problemleri veya diğer hataların etkisi de çok daha ciddi olmaktadır.

Gelinen noktada, böylesine karmaşık ve yazılım yoğun sistemlerin doğrulanması için yazılım testi ihtiyacı her zamankinden daha fazla hissedilmektedir. Yazılım testi, web uygulamalarının işlevsellik, güvenlik ve performans beklentilerini karşıladığını garanti altına alarak hem kullanıcı memnuniyetini hem de sistem güvenilirliğini sağlamada kritik bir rol oynamaktadır. (Kertusha ve diğerleri, 2025b: 3). Literatürde web uygulaması testinin, bu tür uygulamaların fonksiyonellik, güvenlik ve güvenilirliğini temin etmede vazgeçilmez olduğu vurgulanmaktadır (Li ve diğerleri, 2018).

Test süreçleri temelde manuel testler ve otomasyon testleri olarak ikiye ayrılmaktadır. Manuel testler, doğrudan insan faktörüne dayalıdır ve bir test uzmanının tecrübesi, dikkati ve bilgi birikimi doğrultusunda gerçekleştirilir ancak bu durum,

manuel testlerin tutarlılık ve hız açısından zayıf kalmasına yol açar. Birçok negatif yönü ile birlikte manuel test süreçleri, tekrarlanabilirlikten uzak olup hata riskini artırmakta ve büyük projelerde zaman ve maliyet açısından verimsiz hale gelmektedir. Otomasyon testleri ise insan bağımlılığını ortadan kaldırarak testlerin tutarlı, hızlı ve tekrarlanabilir bir şekilde gerçekleştirilmesini sağlar. Test senaryolarının otomatikleştirilmesi, manuel testlerde karşılaşılan hata oranlarını minimuma indirirken, aynı zamanda zaman ve maliyet tasarrufu sağlamaktadır. Özellikle büyük ölçekli ve sık güncellemeye ihtiyaç duyan web uygulamalarında, otomasyon testleri sürekli entegrasyon/sürekli dağıtım (Continuous Integration Continuous Delivery: CI/CD) süreçleriyle entegre edilerek yazılım kalitesinin sürdürülebilirliğini artırmaktadır (coderspace.io, 2024).

Çalışma, web tabanlı yazılım geliştirme süreçlerinde yaygın olarak kullanılan test otomasyon araçlarını, çok kriterli karar verme yöntemleri kullanarak seçim kriterlerinin önem sırasına konulmasına ve araçların karşılaştırılmasına odaklanmaktadır. Çalışmanın temel amacı ise; yazılım geliştiriciler, test mühendisleri ve karar vericiler için en uygun test otomasyon aracının seçim sürecini kolaylaştırmak ve bu alanda daha bilinçli kararlar alınmasına katkı sağlamaktır. Doğru otomasyon aracı seçmek otomasyon testi sürecinin başarısına direkt etki etmektedir. (Harmancı ve diğerleri, 2025)

Çalışmada karşılaştırılacak araçlar Selenium, Cypress, Puppeteer ve Playwright olarak belirlenmiş olup, bu araçlar açık kaynaklı, en çok indirilen ve web testi özelinde geliştirilmiş olmaları nedeniyle seçilmiştir. Diğer test otomasyon araçları ise kapsam dışında bırakılmıştır çünkü çalışma kapsamında indirme sayısı, erişilebilirlik, topluluk desteği, açık kaynaklı olması gibi faktörler ve araçların yaygın kullanımı göz önünde bulundurulmuştur. Ayrıca, karşılaştırmanın derinlemesine ve tutarlı olması için belirli sayıda araç üzerinde odaklanmak bir gereklilik olarak görülmüştür. Belirlenen kısıtlar doğrultusunda yapılan seçimler çalışma süresince daha kapsamlı bir analiz yapılmasına olanak sağlamaktadır.

Araçların birbirleriyle karşılaştırılıp sıralanması için Çok kriterli karar verme (Multi Criteria Decision Making: MCDM, ÇKKV) yöntemlerinden yararlanılmıştır. ÇKKV yöntemlerinden olan Geliştirilmiş Bulanık SWARA ve Bulanık MARCOS yöntemleri, sırasıyla kriter ağırlıklandırma ve dört alternatifin sıralanması amacıyla

tez içerisinde kullanılmak üzere tercih edilmiştir. Söz konusu yöntemlerin tercih edilmesinin iki temel nedeni bulunmaktadır. İlk olarak, seçilen yöntemlerin diğer ÇKKV yöntemlerine kıyasla daha az matematiksel hesaplama gerektirmesi ve uygulama süreçlerinin daha pratik olması tercih edilmesinde önem taşımıştır (Ghorabaee ve diğerleri (2018) , Stević ve diğerleri (2020)). İkinci olarak, her iki yöntem de sübjektif değerlendirmeleri ve bulanık mantığı temel alarak uzman görüşlerinin daha doğru ve mantıksal şekilde işlenmesine olanak tanımaktadır. Böylelikle, karar verme sürecinde belirsizliklerin etkisi azaltılarak daha güvenilir sonuçlara ulaşılması hedeflenmiştir.

Araçların karşılaştırılabilmesi amacıyla, literatürde ilgili alanda gerçekleştirilen çalışmalar temel alınarak yedi farklı kriter belirlenmiştir. Kriterlerin her bir araç özelinde değerlendirilmesi için on adet uzman test otomasyon mühendisi ile odak görüşmesi gerçekleştirilmiştir. Uzmanların görüşleri doğrultusunda elde edilen veriler analiz edilerek araçların karşılaştırma sonuçları ortaya konmuştur.

Tezin ilk bölümünde, literatür taraması kapsamında test otomasyon araçları üzerinde gerçekleştirilen ulusal ve uluslararası çalışmalar incelenmiştir. Çalışmaların amacı ve konusu özetler halinde sunularak çalışmaların mevcut yaklaşımları ve bulguları derlenmiştir. Bölümün sonunda ise ele alınan çalışmalar tablo üzerinde sunulmuştur.

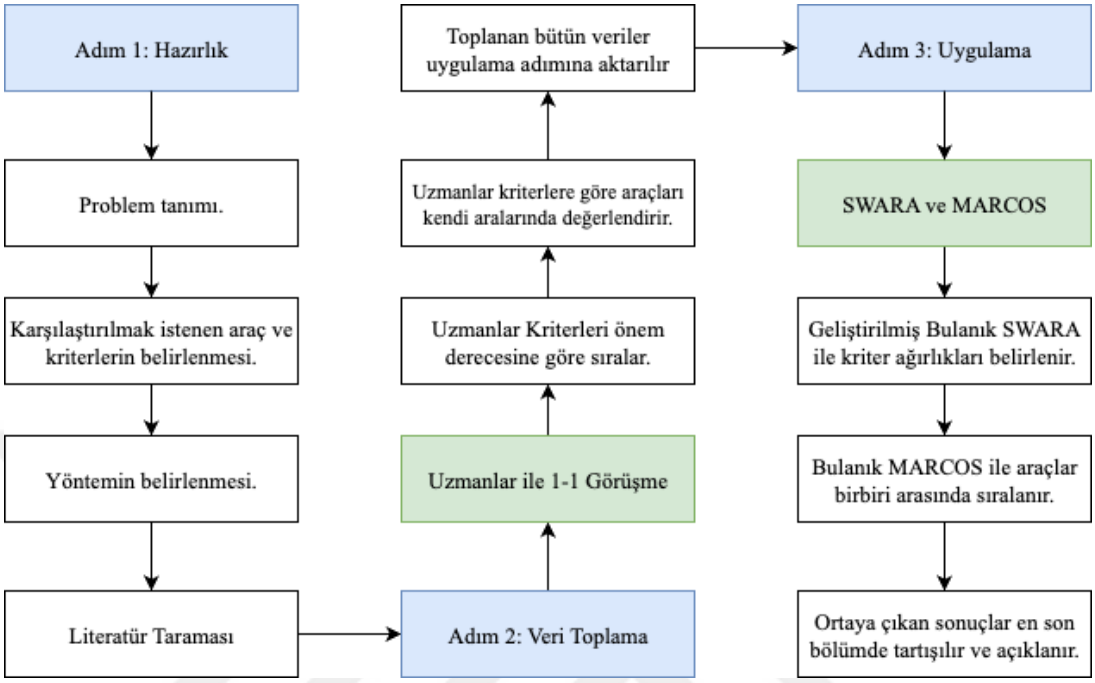
İkinci bölümünde, teknoloji ve internetin gelişimiyle birlikte hayatımıza giren web uygulamaları ve yazılım testi kavramlarına yer verilmiştir. Web'in tarihsel gelişim süreci ele alınarak, web yazılımlarının giderek daha karmaşık hale gelmesi ve bu durumun yazılım testi ihtiyacını nasıl ortaya çıkardığı incelenmiştir. Ayrıca, yazılım test süreçlerinde yaygın olarak kullanılan manuel test ve test otomasyonu kavramları açıklanmıştır. Tez kapsamında karşılaştırılacak dört farklı otomasyon aracının tanıtımı yapılmış; bu araçların ortaya çıkış süreci, kullanım alanları, yapısı ve öne çıkan özellikleri detaylı bir şekilde ele alınmıştır.

Tezin üçüncü bölümünde, çok kriterli karar verme (ÇKKV) yöntemlerinin kavramsal çerçevesi açıklanmış ve bu yöntemlerin temel prensipleri ele alınmıştır. Tez kapsamında kullanılacak olan Geliştirilmiş Bulanık SWARA ve Bulanık MARCOS yöntemleri ayrıntılı şekilde incelenmiştir. Ek olarak, bulanık mantık yaklaşımı tanıtılarak bu yaklaşımın ÇKKV yöntemleriyle entegrasyonunun sağladığı avantajlar üzerinde durulmuştur.

Tezin dördüncü bölümü içerisinde test uzmanlarından alınan veriler sırasıyla Geliştirilmiş Bulanık SWARA ve Bulanık MARCOS yöntemleriyle analiz edilmiştir. İki farklı ÇKKV yöntemi içinde yer alan adımlar detaylı biçimde tablolaştırılarak sunulmuştur.

Son olarak tezin sonuç bölümünde, tezin amacı, neden teze ihtiyaç olunduğu, literatüre katkısı, daha önceki çalışmalardan farkı, gelecek çalışmalara sunduğu fırsatlar ve yapılan analizlerin sonucu incelenmiştir. Şekil 1'de tezin kavramsal çerçevesi sunulmuştur.

Şekil 1: Kavramsal Çerçeve



Kaynak**:** Yazar tarafından derlenmiştir.

Şekil 1 üzerinde görüldüğü gibi hazırlık aşaması tezin en fazla adımı olan ve uzun süren aşamasıdır. Diğer aşamalar uzmanlarla süren kısa görüşmelerle ve çok kriterli karar verme uygulamalarını uygulamakla geçmiştir.

BİRİNCİ BÖLÜM
LİTERATÜR ARAŞTIRMASI

Tezin birinci bölümü literatürde test otomasyon araçları ile ilgili yapılmış olan çalışmaları açıklamayı hedeflemektedir. Özel olarak web alanında yapılan çalışmaların yanı sıra genel anlamda otomasyon testi araçları ile ilgili yapılan karşılaştırmalar ve anketler de bölüm içerisine eklenmiştir çünkü çalışma içerisinde incelenecek araçlar web harici otomasyon testlerini de gerçekleştirebilir.

1.1. LİTERATÜR ARAŞTIRMASI

Literatürde birçok farklı araştırmanın ilgili alanda yapıldığı gözlemlenmektedir ilk olarak; Garousi, Coşkunçay, Betin Can ve Demirörs (2013), Türkiye'deki yazılım test uygulamalarını incelemek amacıyla geniş kapsamlı bir anket çalışması gerçekleştirmiştir. Çalışmanın amacı, Türk yazılım endüstrisindeki test süreçlerini, kullanılan test tekniklerini, otomasyon araçlarını ve karşılaşılan zorlukları tespit etmektir. Araştırma kapsamında, Kanada ve SWEBOK rehberliğinde tasarlanan, 9'u yazılım testine odaklanan toplam 54 soruluk bir anket uygulanmış ve Türkiye'den 163 yazılım mühendisi katılım sağlamıştır. Anket sonuçlarına göre; fonksiyonel/sistem testi, kullanıcı kabul testi ve entegrasyon testi en yaygın kullanılan test türleri olurken, stres testi en az tercih edilen teknik olmuştur. Türkiye'de kod kapsama oranı gibi ölçütlerin yaygın olmadığı, test ve kalite ölçütlerinin düşük seviyede kullanıldığı tespit edilmiştir. Katılımcıların çoğunluğu manuel test yöntemlerini tercih etmekte olup, otomasyon kullanımı daha sınırlıdır. Ayrıca, test mühendisleri ile geliştiriciler arasındaki oran genellikle 1:2 ile 1:5 arasında değişmektedir. Çalışma, Türkiye'deki yazılım test kültürünün henüz gelişim aşamasında olduğunu ortaya koymuş ve test süreçlerinin daha sistematik hale getirilmesi için akademi ve endüstri iş birliğinin önemini vurgulamıştır.

Shaheen, Rauf ve Hameed (2013), Pakistan'daki yazılım endüstrisinde yazılım test otomasyonu (AST) uygulamalarını inceleyen bir endüstriyel araştırma gerçekleştirmiştir. Çalışma, AST'nin faydalarını ve zorluklarını belirlemek,

literatürdeki bilgilerle yerel uygulamalar arasındaki boşlukları keşfetmek ve sektörde kullanılan otomasyon araçlarını değerlendirmek amacıyla tasarlanmıştır. Araştırma bulguları, AST'nin Pakistan'da son yıllarda tanıtıldığını ve uygulama sürecinin genellikle yeni olduğunu göstermektedir. En sık kullanılan test araçlarının HP QuickTest Professional ve Selenium olduğu belirtilmiştir. AST'nin zaman tasarrufu, maliyet azaltma ve kalite iyileştirme gibi avantajları vurgulanmış ancak bakım maliyetlerinin yüksekliği, karmaşık test senaryolarının yönetimi ve eğitim gereksinimlerinin zorluk yarattığı belirtilmiştir. Çalışma, otomasyonun yaygınlaştırılması için araştırma ve geliştirme çalışmalarına ihtiyaç duyulduğunu ortaya koymuştur.

Sharma ve Angmo (2014), web tabanlı otomasyon test araçlarının analizi ve bunların yazılım geliştirme süreçlerinde nasıl kullanıldığını incelemiştir. Çalışma, web tabanlı uygulamaların test edilmesinin maliyeti azaltarak ve test sürecini hızlandırarak kaliteyi artırabileceğini vurgulamaktadır. Selenium, HP-QTP, FitNesse, Watir, TestComplete ve LoadRunner gibi popüler araçlar ele alınmış, bu araçların özellikleri, tarayıcı ve platform desteği ile kullanım senaryoları değerlendirilmiştir. Özellikle Selenium'un açık kaynaklı bir araç olarak geniş bir kullanıcı kitlesine hitap ettiği ve farklı tarayıcılar arasında kolayca entegre olabildiği belirtilmiştir. Çalışmada, doğru otomasyon aracını seçmenin test süreçlerinin etkinliğini artırdığı ve uygulama gereksinimleriyle uyumlu bir aracın seçilmesinin önemli olduğu ifade edilmiştir. Sonuç olarak, web tabanlı otomasyon testi ve bu tür testler için kullanılan araçlar üzerine yapılan bu çalışmaya göre, Selenium'un şu ana kadar web uygulamaları için en iyi otomasyon aracı olduğu belirtilmiştir.

Abbas, Sultan ve Bhatti (2017), dört otomasyon yük testi aracını (Apache JMeter, HP LoadRunner, Microsoft Visual Studio (TFS) ve Siege) karşılaştırmalı olarak analiz eden bir çalışma gerçekleştirmiştir. Araştırma, bu araçları test senaryoları oluşturma, eklenti desteği, sonuç raporlaması, uygulama desteği ve maliyet gibi kriterlere göre değerlendirmiştir. Çalışma sonuçları, her aracın farklı avantaj ve dezavantajlar sunduğunu göstermektedir. JMeter, ücretsiz oluşu ve geniş eklenti desteği ile öne çıkarken, LoadRunner yüksek maliyeti ve karmaşık yapılandırma gereksinimleriyle dikkat çekmiştir. Siege, basit komut satırı arayüzü ve hızlı sonuçlarıyla öne çıkmasına rağmen sınırlı seçenekler sunmaktadır. TFS ise kullanıcı

dostu arayüzü ve kapsamlı raporlama özellikleriyle dikkat çekmiş, ancak yalnızca Windows ortamını desteklemesi bir dezavantaj olarak belirtilmiştir. Çalışma, doğru yük testi aracını seçerken bütçe, test edilen yazılımın doğası ve kullanıcı gereksinimlerinin dikkate alınması gerektiğini vurgulamaktadır. Gelecek çalışmalarda, özellikle stres testleri sırasında araçların performansının daha ayrıntılı incelenmesi önerilmiştir.

Sabev ve Grigorova (2017), yazılım testi için en popüler grafiksel kullanıcı arayüzü (GUI) otomasyon araçlarını karşılaştırmalı bir analizle değerlendirmiştir. Çalışmada, 52 farklı GUI otomasyon aracı incelenmiş, bu araçlar popülerlik, olgunluk, maliyet, kullanılabilirlik ve raporlama gibi sekiz farklı kritere göre değerlendirilmiştir. Son aşamada, 10 araç ayrıntılı bir analize tabi tutulmuştur. En yüksek puanı alan araçlar Unified Functional Testing (UFT) ve Selenium olmuştur. Çalışma, GUI test otomasyon araçlarının seçiminde, kullanıcı dostu olması, entegrasyon yetenekleri ve test senaryolarının sürdürülebilirliği gibi kriterlerin önemini vurgulamaktadır. Araçların olgunluk seviyesi ve topluluk desteği, seçimi etkileyen önemli faktörler arasında yer almıştır. Çalışma, endüstri ve akademi için kapsamlı bir araç seçim rehberi sunmaktadır.

Sinaga, Wibowo P., Silalahi ve Yolanda (2018), Android uygulamaları için üç popüler otomasyon testi aracı olan Appium, Robotium ve UI Automator'un performansını karşılaştırmalı olarak incelemiştir. Çalışma, bu araçların dört farklı Android uygulaması üzerinde, 27 kriter ışığında deneysel olarak test edilmesini kapsamaktadır. Araçların kullanıcı arayüzü, müdahale durumları, depolama ve batarya kullanımı, test aktiviteleri ve bağlantı ayarları gibi özellikleri detaylı şekilde analiz edilmiştir. Sonuçlar, Appium'un özellikle uygulama değiştirme ve bağlantı modlarını yönetme gibi özelliklerde üstün olduğunu, ancak emülatör testlerinde yetersiz kaldığını göstermiştir. Robotium, gri kutu testlerini desteklemesiyle öne çıkarken, UI Automator'un daha hızlı test süreleri sunduğu belirtilmiştir. Çalışma, her aracın farklı senaryolara uygun avantaj ve dezavantajlara sahip olduğunu vurgulamış ve test gereksinimlerine bağlı olarak araç seçimi için öneriler sunmuştur. Gelecekteki araştırmalar için daha fazla test kriteri eklenmesi ve emülatör desteğinin iyileştirilmesi önerilmektedir.

Mobaraya ve Ali (2019), modern web uygulamalarının test edilmesi için kullanılan iki popüler otomasyon çerçevesi olan Selenium ve Cypress'i karşılaştırmalı bir çalışma ile analiz etmiştir. Çalışma, AliExpress web sitesinde kullanıcı hesabı oluşturma ve ödeme işlemleri gibi senaryolar için test otomasyon betikleri geliştirerek bu iki aracın performansını ve verimliliğini değerlendirmiştir. Cypress'in, Selenium'a kıyasla dinamik web öğelerinin yüklenmesi sırasında otomatik bekleme süreleri tanımlama avantajına sahip olduğu, bu nedenle test verimliliğini artırdığı belirtilmiştir. Selenium'un ise daha geniş tarayıcı desteği ve güçlü topluluk desteği ile öne çıktığı ifade edilmiştir. Test yürütme süreleri açısından iki araç arasında belirgin bir fark bulunmamakla birlikte, Cypress daha az kod satırı gereksinimi ile test verimliliğini artırmaktadır. Çalışma, test gereksinimlerine göre her iki aracın avantaj ve dezavantajlarını detaylandırarak, karmaşık web uygulamaları için araç seçimi konusunda rehberlik sunmayı amaçlamaktadır. Gelecek çalışmalar için tarayıcı desteğinin artırılması ve kod raporlama özelliklerinin geliştirilmesi önerilmiştir.

Ekrem Erol (2019), çalışmasında yazılım test otomasyonu araçlarının performans, maliyet ve kullanılabilirlik açısından karşılaştırılmasını amaçlamaktadır. Çeşitli donanım özelliklerine sahip bilgisayarlarda ve farklı türde web sitelerinde, günümüzde popüler olan altı test otomasyon aracı (Selenium WebDriver, Katalon Studio, Telerik Test Studio, Ranorex Studio, Squish ve TestingWhiz) üzerinde deneyler yapılmıştır. Araçlar, CPU ve RAM kullanımı, kullanıcı dostu arayüz, dokümantasyon ve eğitim materyalleri gibi kriterler ışığında değerlendirilmiştir. Bulgular, Telerik Test Studio'nun en düşük sistem kaynakları tüketimi ile en etkili performansı sergilediğini, ancak Katalon Studio gibi araçların kolay kurulumu ve ücretsiz olması nedeniyle daha erişilebilir olduğunu göstermektedir. Çalışma, kullanıcıların test otomasyon aracı seçiminde bilinçli kararlar alabilmeleri için kapsamlı bir kaynak oluşturmayı hedeflemektedir.

Umar ve Zhanfang (2019), yazılım test otomasyonu araçları ve çerçeveleri üzerine kapsamlı bir inceleme sunmaktadır. Çalışmada, yazılım projelerinde test otomasyonunun verimliliği artırmak ve maliyeti düşürmek için kritik olduğu vurgulanmıştır. Araştırma, test otomasyonu araçlarını birden fazla kategoriye ayırarak (birim test araçları, fonksiyonel test araçları, performans test araçları vb.) detaylı bir analiz sunmaktadır. Ayrıca, farklı test otomasyon çerçevelerinin (data-driven,

keyword-driven, hybrid framework) avantaj ve dezavantajlarını açıklamaktadır. Çalışma, Selenium, Katalon Studio, TestComplete ve Unified Functional Testing (UFT) otomasyon araçlarını karşılaştırarak bu araçların güçlü ve zayıf yönlerini değerlendirmiştir. Sonuç olarak, otomasyon aracı seçiminin, test gereksinimleri, bütçe ve organizasyondaki teknik becerilere bağlı olarak değişmesi gerektiği belirtilmiştir. Bu çalışma, yazılım test süreçlerini yazılım projelerine dahil etmenin önemini ortaya koymuş ve araç seçiminin organizasyon ve bütçe gibi değişkenlere bağlı olarak yapılması gerektiğini vurgalmıştır.

Heidilyn V. Gamido ve Marlon V. Gamido (2019), otomasyon yazılım test araçlarının özelliklerini karşılaştırmalı olarak inceleyen bir çalışma gerçekleştirmiştir. Çalışma, manuel testlerin zaman alıcı ve hata yapmaya açık olması durumlarında, otomasyon test araçlarının daha etkili bir çözüm sunduğunu vurgulamaktadır. Selenium, QTP/UFT, TestComplete, Ranorex, LoadRunner, Watir, Sahi Pro ve SoapUI gibi araçlar, platform desteği, tarayıcı uyumluluğu, programlama dili desteği, kayıt-oynatma özellikleri, öğrenme kolaylığı, raporlama yetenekleri ve maliyet gibi kriterlere göre analiz edilmiştir. Çalışmanın sonuçlarına göre, proje maliyeti öncelikli bir kriterse Selenium gibi açık kaynaklı araçlar tercih edilebilirken, destek hizmetleri, kolay öğrenim ve raporlama özellikleri öncelikli olduğunda QTP/UFT gibi lisanslı araçlar daha uygun bir seçenek sunmaktadır. Gelecek çalışmalar için, daha fazla test aracının yanı sıra bu araçların yanıt sürelerini de içeren kapsamlı bir analiz önerilmektedir.

Al Sawaf (2019), web tabanlı otomatik test araçlarının değerlendirilmesi ve karşılaştırılması üzerine bir yüksek lisans tezi hazırlamıştır. Çalışma, web uygulamalarının karmaşıklığının artmasıyla birlikte, bu uygulamaları test etmek için kullanılan otomatik test araçlarını kapsamlı bir şekilde analiz etmeyi amaçlamaktadır. Selenium, TestComplete, JMeter, Webload, TestingWhiz ve Acunetix gibi araçlar, tarayıcı uyumluluğu, teknik destek, maliyet, kullanım kolaylığı ve performans kriterleri doğrultusunda karşılaştırılmıştır. Sonuçlar, Selenium'un açık kaynaklı yapısı ve geniş kullanım olanakları nedeniyle öne çıktığını, ancak ticari araçların, özellikle TestComplete ve Webload'un, daha kapsamlı özellikler ve teknik destek sağladığını göstermektedir. Çalışmada, güvenlik testleri için Acunetix, kullanılabilirlik testleri için GTmetrix ve performans testleri için JMeter'ın önerildiği belirtilmiştir. Gelecek

çalışmalar için, otomasyon test süreçlerine yapay zekâ entegrasyonunun, test senaryolarının oluşturulmasını ve yönetimini daha verimli hale getirebileceği önerilmektedir.

Melkozerova, Nariezhnii ve Malakhov (2019), yazılım test otomasyonu araçlarını analiz ederek bu araçların özelliklerini, teknolojilerini, kullanım alanlarını ve avantajlarını kapsamlı bir şekilde incelemiştir. Çalışma, Selenium, Serenity BDD ve JMeter gibi araçları örnek alarak test süreçlerini optimize etmek için bu araçların nasıl kullanılabileceğini göstermektedir. Araştırmada, veri ve anahtar kelime odaklı testler, kayıt ve oynatma (record and playback) teknolojileri gibi test otomasyon teknikleri ele alınmıştır. Özellikle Selenium WebDriver'ın çapraz tarayıcı desteği ve karmaşık test senaryolarını destekleme kapasitesine vurgu yapılmıştır. Serenity BDD'nin davranış tabanlı test senaryolarında kullanıcı dostu bir yaklaşım sunduğu belirtilmiş, JMeter'ın ise performans testi için güçlü bir çözüm olduğu ifade edilmiştir. Çalışma, otomasyon araçlarının seçiminde, test gereksinimlerine ve projenin doğasına uygun araçların kullanılmasının önemini vurgulamaktadır.

Alferidah ve Ahmed (2020), yazılım test otomasyonu araçlarının sınıflandırılması ve karşılaştırılmasını ele alan bir çalışma sunmuştur. Çalışma, yazılım sistemlerinin kalite, güvenilirlik ve performansını artırmak için otomasyon araçlarının önemini vurgulamaktadır. Araştırmada, işlevsel test, yük testi, regresyon testi ve uçtan uca test gibi farklı kategorilerdeki araçlar analiz edilmiştir. Apache JMeter, yük testi için doğruluk ve esneklik sağlayan açık kaynaklı bir araç olarak tanımlanırken, LoadRunner performans analizi için güçlü bir araç olarak değerlendirilmiş ancak yapılandırma zorluklarına dikkat çekilmiştir. Selenium, platformlar arası desteği ve kolay kurulumu sayesinde en popüler işlevsel ve regresyon test araçlarından biri olarak belirtilmiştir. Ayrıca, Robotium'un uçtan uca testler için etkili bir çözüm sunduğu ifade edilmiştir. Çalışma, test otomasyon araçlarının verimliliğini artırmak ve yazılım geliştirme süreçlerini optimize etmek için kullanım kolaylığı, öğrenim süreci ve sistem uyumluluğu gibi kriterlerin önemli olduğunu vurgulamıştır. Gelecekte, test otomasyonu araçlarının kullanımının artacağı ve daha geniş bir benimsenme göreceği öngörülmüştür.

Ayşe Kaymak (2020), yazılım test otomasyon araçlarının karşılaştırılması ve manuel test yerine otomasyon seçiminin gerekliliği üzerine kapsamlı bir yüksek lisans

tezi hazırlamıştır. Çalışma, yazılım testi sürecinde otomasyonun etkinliğini artırmak ve doğru aracı seçmenin önemini vurgulamaktadır. Telerik Test Studio, Katalon Studio, Sahi Pro ve Selenium gibi popüler test araçları, test sonuç raporları, öğrenim kolaylığı, kurulum zorluğu, tarayıcı desteği, platform uyumluluğu ve hız gibi 10 farklı kriter üzerinden değerlendirilmiştir. Her araç, belirli bir test senaryosunda kullanılarak avantajları ve dezavantajlarıyla analiz edilmiştir. Sonuçlar, Sahi Pro'nun birçok kategoride en yüksek puanı aldığını ve daha az bilinen araçların bile uygun senaryolarda etkili olabileceğini göstermiştir. Çalışma, yazılım mühendislerine projelerine uygun test araçlarını seçmede rehberlik etmeyi amaçlamaktadır ve gelecek araştırmalar için detaylı bir karşılaştırma metodolojisi sunmaktadır.

Pelivani ve Cico (2021), web tabanlı uygulamalar için kullanılan otomasyon test araçlarını değerlendirmiş ve Katalon Studio ile Selenium'u çeşitli açılardan karşılaştırmıştır. Çalışmada, iki aracın kurulum kolaylığı, raporlama yetenekleri, test süresi, entegrasyon kabiliyeti ve programlama dili desteği gibi özellikleri analiz edilmiştir. Katalon Studio, kullanıcı dostu arayüzü ve hızlı test oluşturma kabiliyeti ile öne çıkarken, Selenium açık kaynaklı olması ve çoklu dil desteği ile tercih edilmektedir. Ancak Selenium'un raporlama desteğinin sınırlı olması ve kurulumunun daha karmaşık olduğu belirtilmiştir. Çalışma, doğru otomasyon aracını seçmenin yazılım testi projelerinin başarısındaki önemini vurgulamakta ve bu araçların güçlü ve zayıf yönlerini detaylı bir şekilde sunarak karar alma sürecine rehberlik etmektedir.

Pelivani, Besimi ve Cico (2022), dört popüler kullanıcı arayüzü (UI) test otomasyon çerçevesi olan Selenium, Cypress, WebDriverIO ve Robot Framework'ü karşılaştırmalı bir çalışmayla değerlendirmiştir. Çalışma, bu çerçevelerin kurulum kolaylığı, öğrenme eğrisi, entegrasyon yetenekleri, test süresi ve raporlama özelliklerini analiz etmektedir. Selenium, geniş dil desteği ve çapraz platform uyumluluğu ile öne çıkarken, Cypress hız ve hata tahmini özellikleriyle dikkat çekmiştir. WebDriverIO, hızlı çalışması ve mobil uyumluluğu sayesinde etkili bulunmuştur. Robot Framework ise programlama bilgisi gerektirmeyen anahtar kelime tabanlı yapısı sayesinde kullanıcı dostu bir seçenek sunmaktadır ancak daha yavaş test süresiyle eleştirilmiştir. Çalışma, doğru test çerçevesi seçiminin proje ölçeği, bütçe ve kullanılan teknolojiye bağlı olduğunu vurgulamakta ve sektörel uzmanlar için kapsamlı bir rehber sunmaktadır.

Samli ve Orman (2023), yazılım test otomasyonu araçlarının web tabanlı uygulamalar üzerindeki performansını değerlendiren kapsamlı bir çalışma yapmıştır. Bu çalışmada, ücretsiz ve lisanslı olmak üzere toplam 14 araç (Selenium, Katalon Studio, JMeter, LoadRunner, Telerik Test Studio, Ranorex Studio, TestComplete, ve diğerleri) 20 farklı kriter doğrultusunda karşılaştırılmıştır. Araçlar, CPU ve RAM gereksinimleri, programlama dili uyumluluğu, platform desteği, kullanıcı deneyimi ve hata takibi gibi özellikler açısından incelenmiştir. Bulgular, her aracın farklı senaryolarda öne çıkan avantaj ve dezavantajlara sahip olduğunu göstermektedir. Örneğin, Acunetix güvenlik açıklarının tespitinde başarılı bulunurken, Selenium çok yönlülüğüyle dikkat çekmiştir ancak raporlama yeteneklerinin sınırlı olduğu belirtilmiştir. Çalışma, yazılım test uzmanlarının projelerine uygun araçları seçmelerine rehberlik etmek amacıyla detaylı bir kaynak sağlamayı hedeflemiştir.

Ali Aburas (2024), yazılım test otomasyonu araçlarının seçimi üzerine kapsamlı bir çalışma gerçekleştirmiştir. Çalışmada, yazılım projelerinde kaliteyi artırırken maliyeti ve süreyi azaltmanın yollarını araştırmak amacıyla otomasyon test araçları değerlendirilmiştir. Toplamda 16 popüler test otomasyon aracı, platform uyumluluğu, desteklediği test türleri (örneğin birim testi, entegrasyon testi ve sistem testi) ve hedef programlama dilleri gibi kriterlere göre incelenmiştir. Selenium, Appium ve Robot Framework gibi araçlar, çok yönlülükleri ve farklı platformları destekleme kabiliyetleri nedeniyle öne çıkmıştır. Ayrıca JUnit ve EvoSuite gibi araçlar, Java tabanlı birim testlerinde güçlü performans göstermiştir. Çalışma, otomasyon araçlarının doğru seçiminin test süreçlerini optimize etmek için kritik olduğunu vurgulamakta ve yazılım geliştiricilere, test mühendislerine ve araştırmacılara rehberlik etmektedir. Araştırma, araçların açık kaynaklı veya ticari lisanslı olup olmalarına göre avantaj ve dezavantajlarını da değerlendirmiştir. Sonuç olarak, projenin gereksinimlerine, hedef platformlara ve kodlama diline bağlı olarak en uygun aracın seçilmesi gerektiği belirtilmiştir.

1.2. LİTERATÜR ÖZET

Literatüre kazandırılan yayınlar incelendiğinde web özelinde test otomasyon araçlarının seçildiği ve karşılaştırıldığı birçok değerli çalışmanın mevcut olduğu gözlemlenmektedir. Mevcut çalışma ise diğer yapılan yayınların içeriğine konu anlamında benzerlik gösterse de özellikle en çok indirilme sayısına sahip araçların seçimi, seçilen araçların web otomasyon testi özelinde oluşturulması ve karma çok kriterli karar verme yaklaşımının tez yöntemi olarak kullanılması ile diğer çalışmalardan farklılaştığı kısımlardır.

Tablo 1: Literatür Araştırmalarına Dair Özet Tablo

Yazar / Yazarlar	Basım Yılı	Karşılaştırılan Araçlar	Yöntem	Konu
Garousi, Coşkunçay, Betin Can ve Demirörs	2013	HP QuickTest Professional, Selenium	Anket	Türkiye'deki yazılım test uygulamalarının analizi
Shaheen, Rauf ve Hameed	2013	HP QuickTest Professional, Selenium	Karılaştırılmalı analiz	Pakistan'daki otomasyon test uygulamalarının analizi
Sharma ve Angmo	2014	Selenium, HP-QTP, FitNesse, Watir, TestComplete, LoadRunner	Karşılaştırmalı analiz	Web tabanlı test araçlarının karşılaştırılması
Abbas, Sultan ve Bhatti	2017	Apache JMeter, HP LoadRunner, Microsoft Visual Studio (TFS), Siege	Karşılaştırmalı analiz	Yük testi araçlarının performans analizleri
Sabev ve Grigorova	2017	Unified Functional Testing (UFT), Selenium	Karşılaştırmalı analiz	GUI test otomasyon araçlarının karşılaştırılması
Sinaga, Wibowo, Silalahi ve Yolanda	2018	Appium, Robotium, UI Automator	Deneysel analiz	Android uygulamaları için mobil test araçlarının performans değerlendirmesi
Mobaraya ve Ali	2019	Selenium, Cypress	Performans Testi	Selenium ve Cypress'in performans karşılaştırması
Ekrem Erol	2019	Selenium WebDriver, Katalon Studio, Telerik Test	Deneysel analiz	Test araçlarının CPU, RAM ve kullanılabilirlik

		Studio, Ranorex Studio, Squish, TestingWhiz		açısından karşılaştırılması
Umar ve Zhanfang	2019	Selenium, Katalon Studio, TestComplete, UFT	Kapsamlı inceleme	Test otomasyonu çerçevelerinin analizi
Heidilyn V. Gamido ve Marlon V. Gamido	2019	Selenium, QTP/UFT, TestComplete, Ranorex, LoadRunner, Watir, Sahi Pro, SoapUI	Karşılaştırmalı analiz	Otomasyon test araçlarının karşılaştırılması
Al Sawaf	2019	Selenium, TestComplete, JMeter, Webload, TestingWhiz, Acunetix	Karşılaştırmalı analiz	Web tabanlı test araçlarının değerlendirilmesi
Melkozerova, Nariezhnii ve Malakhov	2019	Selenium, Serenity BDD, JMeter	Teknik analiz	Test araçlarının özellik ve kullanım alanlarının incelenmesi
Alferidah ve Ahmed	2020	Apache JMeter, LoadRunner, Selenium, Robotium	Karşılaştırmalı analiz	Yazılım test otomasyon araçlarının performans analizi
Ayşe Kaymak	2020	Telerik Test Studio, Katalon Studio, Sahi Pro, Selenium	Karşılaştırmalı analiz	Test araçlarının kurulum, raporlama ve hız açısından karşılaştırılması
Pelivani ve Cico	2021	Katalon Studio, Selenium	Performans analizi	Web tabanlı test araçlarının karşılaştırılması
Pelivani, Besimi ve Cico	2022	Selenium, Cypress, WebDriverIO, Robot Framework	Karşılaştırmalı analiz	Kullanıcı arayüzü test araçlarının karşılaştırılması
Samli ve Orman	2023	Selenium, Katalon Studio, JMeter, LoadRunner, Telerik Test Studio, Ranorex Studio, TestComplete	Karşılaştırmalı analiz	Test araçlarının CPU ve RAM açısından incelenmesi
Ali Aburas	2024	Selenium, Appium, Robot Framework, JUnit, EvoSuite	Karşılaştırmalı analiz	Otomasyon araçlarının seçimi ve performans analizi

Kaynak: Yazar tarafından derlenmiştir.

İKİNCİ BÖLÜM
WEB UYGULAMALARI VE TEST OTOMASYON

Geçmişten günümüze insanlar, bilgiyi aktarmak ve iletişim sağlamak amacıyla pek çok araçtan yararlanmış, matbaadan telgrafa, radyodan televizyona kadar farklı teknolojiler bu iletişim ihtiyacını karşılamada önemli roller üstlenmiştir. Ancak 20. yüzyılın son çeyreğinde ortaya çıkan internet teknolojileri, bilgi paylaşımı ve iletişim süreçlerini tümüyle dönüştürmüş, dijital devrimin temelini oluşturmuştur (Castells, 2010). İnternetin dünya çapında yaygınlaşmasıyla birlikte web tabanlı platformların kullanımı, bilgi paylaşımından ticarete, eğitimden eğlenceye kadar günlük hayatın hemen her alanında etkisini göstermiştir (Dijck ve diğerleri, 2018). Bu dijital dönüşüm, web yazılımlarının daha karmaşık, dinamik ve etkileşimli hale gelmesine neden olmuş ve yazılım sistemlerinin kalitesini, güvenilirliğini ve performansını güvence altına almak amacıyla kapsamlı ve etkin yazılım testi süreçlerinin önemini artırmıştır (Garousi ve Mäntylä, 2016). Bu kapsamda birinci bölümde web uygulamaları ve gelişim tarihi, yazılım testi ve test otomasyon araçları konuları ele alınmıştır.

2.1. WEB UYGULAMASI

Teknolojinin ve internet altyapısının gelişimine paralel olarak, yazılım sistemleri de sürekli güncellenmiş ve web tabanlı çözümler giderek daha dinamik ve yenilenebilir bir yapıya kavuşmuştur. Eğitim, sağlık, lojistik ve finans gibi pek çok sektörde yaygın biçimde kullanılan web uygulamaları, literatürde farklı tanımlamalarla ele alınmaktadır.

Web uygulamaları, uzaktaki bir sunucuda barındırılan ve kullanıcıların internet tarayıcıları aracılığıyla erişim sağladıkları yazılım programlarıdır. Uygulamalar, bireysel kullanıcıların ya da kurumsal yapıların ihtiyaçlarına göre geliştirilmekte olup, çok çeşitli amaçlar için kullanılmaktadır. E-posta servisleri, çevrimiçi hesap makineleri, sosyal medya platformları ve e-ticaret siteleri, yaygın kullanılan web uygulamalarına örnek olarak gösterilebilir. Bazı web uygulamaları yalnızca belirli tarayıcılarda çalışırken, çoğu uygulama tüm tarayıcılardan erişilebilir durumdadır. Geleneksel masaüstü yazılımlarından farklı olarak, web uygulamaları; akıllı telefon,

tablet veya masaüstü bilgisayar gibi tarayıcıya sahip herhangi bir cihaz üzerinden kullanılabilmektedir (techtarget.com, 2024).

Web yazılımları, internet veya intranet gibi bir ağ üzerinden, web tarayıcıları kullanılarak erişilebilen uygulamalardır ve genellikle JavaScript, Java, C# gibi çeşitli programlama dilleri kullanılarak geliştirilen bilgisayar yazılımlarından oluşmaktadırlar (Clarke, 2012: 23).

Web uygulamaları, web tarayıcıları aracılığıyla dinamik olarak içerik üreten ve genellikle kullanıcı etkileşimleri, uygulama verileri ya da çeşitli bilgiler (örneğin, kullanıcının konumu veya güncel saat gibi) doğrultusunda çalışan yazılımlardır. Bu uygulamalar, bireylerin diğer kullanıcılarla veya işletmelerle etkileşim kurmasını sağlayan en yaygın dijital çözümler arasında yer almaktadır. Özellikle sosyal medya platformları (WordPress, Facebook, Twitter gibi) ile bankacılık, seyahat ve e-ticaret hizmetleri web yazılımları arasında öne çıkmaktadır. Web uygulamalarının, internet bağlantısına ve tarayıcıya sahip herhangi bir kullanıcı tarafından günün her saatinde erişilebilir olması, onları kullanıcı etkileşimleri için vazgeçilmez hale getirmektedir (Sampath ve Sprenkle, 2016: 155).

Belirtilen kaynaklardan hareketle, web uygulamaları 'internet veya intranet üzerinden erişilebilen, tarayıcı tabanlı ve kullanıcı etkileşimlerini esas alan dinamik yazılımlar' olarak tanımlanabilir. Genellikle uzak bir sunucuda depolanan ve herhangi bir cihaz üzerinden tarayıcı aracılığıyla erişilebilen web uygulamaları, platform bağımsız yapılarıyla geniş bir kullanıcı kitlesine hitap etmektedir. Eğitimden finansa, sosyal ağlardan e-ticarete kadar birçok sektörde aktif olarak kullanılan web uygulamaları, kullanıcıların bilgi alışverişi yapmasına, insanlarla veya işletmelerle etkileşim kurmasına olanak sağlamaktadır. Sahip olduğu özellikler, web uygulamalarını modern teknolojinin en önemli araçlarından biri haline getirmektedir.

2.1.1. Web Uygulamalarının Tarihsel Süreci

Teknolojik gelişmeler doğrultusunda internet, 21. yüzyılda günlük yaşamın vazgeçilmez bir parçası haline gelmiştir. İnternet altyapısının sağladığı olanaklar, sürekli yeni teknolojilerin ortaya çıkmasına zemin hazırlamaktadır. Bu bağlamda, bilgisayar ve tarayıcılar aracılığıyla erişilen ve dünyanın en büyük bilgi ağı olarak

nitelendirilen dünya çapındaki ağ (World Wide Web: WWW), kullanıcıların veri paylaşımı, okuma ve yazma işlemlerini gerçekleştirebildiği temel dijital ortamlardan biridir. Tarihsel süreç içinde, Web 1.0, Web 2.0 ve Web 3.0 dönemleri yaşanmış olup bu teknolojilerin etkisi günümüzde de sürmektedir. Bununla birlikte, yeni nesil web teknolojisi olarak ifade edilen Web 4.0 kavramının literatürde henüz üzerinde uzlaşılmış net bir tanımı bulunmamaktadır (Ersöz, 2020: 58).

Web 1.0 dönemi, WWW kavramının 1990 yılında Tim Berners-Lee'nin öncülüğünde Avrupa Nükleer Araştırma Merkezi'nde (CERN) geliştirilen info.cern.ch adresindeki ilk web sitesinin oluşturulmasıyla hayata geçmiştir. Dönem, internetin geniş kitleler tarafından kullanılmaya başlandığı 1990'lı yılları kapsamaktadır. Web 1.0 sürecinde, web siteleri günümüzdeki dinamik yapıya kıyasla daha statik ve broşür benzeri tasarımlar içermekteydi. Dönem içerisindeki kullanıcılar, yalnızca sunulan içerikleri pasif bir şekilde tüketmekte; herhangi bir veri girişi ya da içerik üretimi gibi etkileşim olanaklarına sahip olmamaktaydılar. Web 1.0 döneminin sahip olduğu karakteristik özellikleri, ilk web tabanlı uygulamaların yapısını anlamak açısından önemli ipuçları sunmaktadır (Kapan ve Öncel, 2020: 278).

- Kullanıcı – Sunucu etkileşimi yok.
- Kullanıcılar verileri yalnızca okuyabilir.
- Statik olarak hazırlanan içeriklerle dolu.
- Azınlıktan çoğunluğa bilgi akışı vardır.

Yukarıda açıklanan dört temel özelliğe göre Web 1.0'da oluşturulan web uygulamalarının statik ve sadece bilgi alma amacıyla oluşturulan basit uygulamalardır.

1999 yılına gelindiğinde Darcy DiNucci isimli bir web tasarımcısının yayınladığı "Fragmented Future" isimli makale içerisinde "Web 2.0" terimi ilk kez kullanılmıştır fakat literatürde yer alan birçok kaynak Web 2.0'ın 2004 yılında O'Reilly Media tarafından düzenlenen bir konferansla literatüre kazandırıldığını belirtmiştir (Toledano, 2013: 6). Web 2.0, kullanıcıların dijital ortamda etkileşim kurarak bilgi alışverişinde bulunmalarına ve çevrim içi topluluklar oluşturmalarına imkân tanıyan yeni nesil web teknolojisi olarak tanımlanabilir (Kırık, 2013: 72). Dönem içerisinde Web 1.0'dan farklı olarak, kullanıcıların içerik üretimine katılımı artmış, site yönetimi azınlığın kontrolünden çıkıp geniş kullanıcı kitlelerine yayılmış ve web siteleri sürekli güncellenen dinamik yapılar haline gelmiştir. Dönem ile birlikte

gelen yeni gelişmeler web sitelerinin düzgün çalışmasını daha karmaşık hale getirmiş ve özellikle yazılım test süreçlerine olan ihtiyacı önemli ölçüde artırmıştır. İki dönem arasındaki farklar, aşağıda sunulan karşılaştırma tablosunda daha net bir biçimde gözlemlenebilmektedir.

Tablo 2: Web 1.0 ve Web 2.0 özellikleri

WEB 1.0 Dönemi	WEB 2.0 Dönemi
Programcıların ürettiği içerik	Kullanıcıların ürettiği içerik
Kişilerden kullanıcılara veri akışı	Kullanıcılardan kullanıcılara veri akışı
Sıkı kontrol mekanizması olan siteler	Az miktarda kontrol edilen siteler
Statik ve az değişen içerik	Dinamik ve sürekli güncellenen içerik
Britannica Online	Wikipedia, Twitter, Facebook

Kaynak: (Işık ve Eşitti, 2015: 14).

Web 3.0, internetin evrim sürecinde önemli bir aşama olarak ortaya çıkmıştır; merkeziyetsizlik, yapay zeka (Artificial Intelligence: AI) entegrasyonu, semantik web ve kullanıcıların verileri üzerindeki kontrolünü güçlendirmesi gibi nitelikleriyle dikkat çekmektedir. Web 3.0 terimini, ilk kez 2006 yılında The New York Times yazarı John Markoff tarafından gündeme getirilmiştir. Web 3.0, birçok açıdan, Tim Berners-Lee tarafından tasarlanan ve herhangi bir merkezi otoriteye bağlı kalmaksızın işleyebilen, kullanıcıların özgürce veri paylaşımında bulunabildiği orijinal web anlayışına dönüş niteliği taşımaktadır (Hypernet, 2022).

Web 3.0 sürecinde dikkat çeken en belirgin yenilikler arasında merkeziyetsizlik ve semantik web kavramları öne çıkmaktadır. Merkeziyetsiz sistemler, Web 1.0 ve Web 2.0 dönemlerinden farklı olarak, içerik ve veri yönetimi üzerindeki kontrolü platform sahiplerinden alıp doğrudan kullanıcılara devretmektedir. Merkeziyetsiz sistem, kullanıcıların verileri üzerindeki yetkilerini artırmakta ve özellikle veri güvenliği ile kişisel mahremiyet açısından önemli kazanımlar sunmaktadır. Diğer taraftan, semantik web kavramının, uygulamaların web ortamındaki verilerin içeriği ve bağlamı ile birlikte anlamlandırması mnasına gelmekte olup anlamlandırma sürecinin meta veriler ve yapay zekâ teknolojilerinin etkin rol alması ile gerçekleştirildiği belirtilmektedir. (Amazon, 2023) Genel anlamda Web 3.0,

veri yönetimi ve kullanıcı deneyiminde önceki dönemlere göre daha ileri bir sistem sunmaktadır.

Günümüzde üç farklı döneme ait uygulamaların hala aktif olarak kullanılmakta olduğu görülmektedir. Bununla birlikte, teknolojik ilerlemeler doğrultusunda nesnelerin interneti (Internet Of Things: IOT), artırılmış gerçeklik (Augmented Reality: AR) ve gelişmiş yapay zeka çözümlerinin yaygınlaşması, Web 4.0 kavramının literatürde giderek daha fazla tartışılmasına zemin hazırlamıştır. Literatürde kesin çizgilerle tanımlanmamış olmakla birlikte, Web 4.0 kavramı; insan-makine etkileşiminin en üst seviyeye ulaştığı, internet altyapısının yapay zeka destekli ultra akıllı bir sisteme evrildiği yeni nesil web anlayışı olarak değerlendirilmektedir. Dönemin temel hedefi, büyük veri (Big Data), yapay zeka, makineler arası iletişim, bulut bilişim ve artırılmış gerçeklik gibi ileri düzey teknolojilerin bütünleşik şekilde kullanılması ile daha akıllı ve etkileşim odaklı bir internet ekosistemi oluşturmaktır. Simbiyotik web, olarak da nitelendirilen Web 4.0; kullanıcılar ile makinelerin gerçek zamanlı, çift yönlü ve akıllı sistemler üzerinden sürekli iletişim kurduğu bir yapıyı öngörmektedir. Bu yeni nesil web anlayışının ayırt edici özelliklerinden biri, nesnelerin interneti teknolojisi ile tam entegrasyon sağlanarak, kullanıcı deneyiminin kişiye özgü hale getirilmesidir. Bir diğer önemli özelliği ise, akıllı ajanlar yardımıyla kullanıcıların çevrim içi etkileşimleri analiz edilmekte ve elde edilen veriler doğrultusunda bireysel ihtiyaçlara yanıt veren hizmetlerin sunulması amaçlanmaktadır. Çalışmalarda, Web 4.0'ın siber-fiziksel sistemlerle destekleneceği, yer ve zamandan bağımsız olarak her türlü cihaz üzerinden erişim sağlanabileceği, ileri düzey otomasyon ve karar destek sistemleriyle donatılacağı ifade edilmektedir. Yapılmak istenen teknolojik dönüşümün, sanayi, sağlık, eğitim, ulaşım ve finans gibi birçok sektörde köklü yapısal değişimlere yol açacağı öngörülmektedir (Ersöz, 2020: 61).

Web teknolojilerinin tarihsel gelişim süreci incelendiğinde, yazılım testine duyulan ihtiyacın, web uygulamalarının yapısal dönüşümüne paralel olarak arttığı görülmektedir. Web 1.0 döneminde, statik ve yalnızca bilgi sunmaya yönelik basit içeriklerden oluşan sistemler yaygınken, kullanıcı etkileşimi oldukça sınırlıydı. Bu nedenle, o dönemde test süreçleri daha az karmaşıktı ve genellikle içerik doğruluğu ile temel erişilebilirlik gibi sınırlı unsurlar kontrol ediliyordu fakat web 2.0 dönemiyle son

kullanıcının içerik üretimine katılımı ve dinamik veri akışı ön plana çıkmış; sürekli güncellenen, etkileşim odaklı ve çok daha kompleks yapıda uygulamalar geliştirilmiştir. Değişen durum, sistemlerin işleyişinde karşılaşılabilecek hata olasılıklarını artırmış ve web sitelerinin tutarlılığını sağlamak için kapsamlı test süreçlerini zorunlu hale getirmiştir. Günümüzde Web 3.0 ve Web 4.0 kavramlarıyla birlikte, merkeziyetsizlik, yapay zeka, nesnelerin interneti gibi ileri teknolojiler ve çok daha kişiselleştirilmiş kullanıcı deneyimleri ön plana çıkmaktadır. Bahsedilen teknolojilerin entegre çalışması, yazılım sistemlerini daha da karmaşık hale getirmiş ve hem işlevsel doğruluk hem de güvenlik testlerinin daha sistematik ve otomatikleştirilmiş yöntemlerle yürütülmesini zorunlu kılmıştır. Dolayısıyla, web'in her evresinde artan kullanıcı etkileşimi, veri çeşitliliği ve sistemler arası entegrasyon gibi faktörler, yazılım testine duyulan gereksinimi sürekli olarak artırmış ve yazılım geliştirme süreci içerisinde yazılım testi gerçekleştirmeyi kaçınılmaz hale getirmiştir.

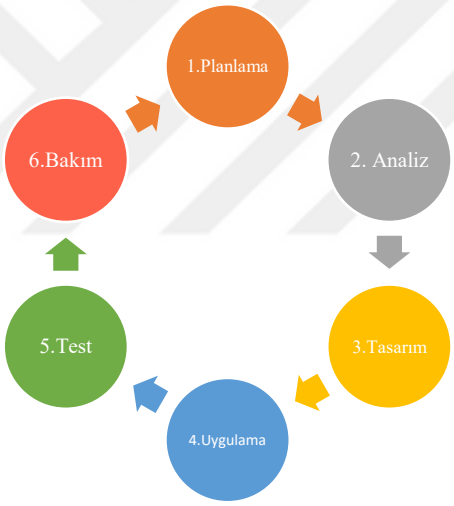
2.2. YAZILIM TESTİ

Günümüzde yazılımlar, özellikle web uygulamaları/yazılımları, bireylerin günlük yaşamından iş dünyasındaki süreçlere kadar geniş bir yelpazede vazgeçilmez hale gelmiştir. Çeşitli sektörlerde dijital dönüşümün temel taşı olan yazılım sistemleri, bireyler açısından da iş başvurularından resmi işlemlere, iletişimden temel ihtiyaçların karşılanmasına kadar pek çok alanda kolaylık sağlamaktadır. Yazılımlara duyulan yoğun bağımlılık, geliştiricilerin daha titiz ve sorumlu hareket etmelerini zorunlu kılmaktadır. Nitekim, hatalı ya da yetersiz yazılımlar, sadece maddi zararlarla sınırlı kalmayıp, kullanıcıların güven kaybına uğramasına ve çeşitli mağduriyetler yaşamasına da yol açabilmektedir. Düşük yazılım kalitesinden kaynaklı sorunlar her yıl 500 milyon dolarlık ekonomik kayba neden olmaktadır. Ayrıca, ABD Ulusal Standartlar ve Teknoloji Enstitüsü (NIST) tarafından yayımlanan raporda, yazılım hatalarının ABD ekonomisine yaklaşık 59,5 milyar dolara mal olduğu, ortaya çıkan maliyetin yaklaşık üçte birinin (22,5 milyar dolar) test süreçlerinin iyileştirilmesi yoluyla azaltılabileceği ifade edilmektedir (Kaymak, 2020). Açıklanan veriler yazılım testi faaliyetlerinin, yazılım geliştirme sürecinin ayrılmaz bir parçası olarak

değerlendirilmesinin gerekliliğini açıkça ortaya koymakta ve yüksek miktarda maddi zararı önlemek amacıyla yapılmasını zorunlu kılmaktadır.

Yazılım testi, yazılım geliştirme yaşam döngüsü (Software Development Life Cycle: SDLC) içerisinde önemli bir aşama olarak değerlendirilmekte olup, yalnızca hata tespitine yönelik bir süreç olmanın ötesinde, yazılımın güvenilirliğini sağlamak, işlevselliğini doğrulamak ve performansını artırmak gibi çok yönlü amaçlar taşımaktadır. Test süreci, yazılımın kullanıcı beklentilerine uygunluğunu denetlemenin yanı sıra potansiyel riskleri azaltarak, sistemin uzun vadeli sürdürülebilirliğine ve güvenlik seviyesinin yükseltilmesine de katkıda bulunmaktadır (coderspace.io, 2024).

Şekil 2. Yazılım Geliştirme Yaşam Döngüsü Adımları



Kaynak: Yazar tarafından derlemiştir.

Literatürde klasik anlamda tasarlanan SDLC modelleri genellikle şekil 2'de olduğu gibi yazılım testini uygulama yapıldıktan sonra yani kod yazım aşamasından sonra başlayan bir süreç olarak belirtmektedir ancak test sürecinin ürünün maliyetine, ortaya çıkma süresine ve başarısına doğrudan etki edecek bir faktör olduğu gözlemlenebilir. Yazılım uygulamasına doğrudan yaptığı etkilerden dolayı yazılım testinin SDLC döngüsünün en başında başlaması gerektiği ve geliştirme sürecinin en önemli aktivitesi olduğu söylenebilir (Kaymak, 2020).

Yazılım testi genel olarak manuel test ve otomasyon testi olmak üzere iki temel yöntemle gerçekleştirilmektedir. Manuel testler, test uzmanlarının yazılımın işleyişini doğrudan gözlemleyerek hataları tespit etmeye çalıştığı subjektif bir yaklaşımı ifade ederken; otomasyon testi, çeşitli yazılım araçlarından faydalanılarak test yapmayı ve test sürecini daha objektif hale getirmeyi amaçlar. Bununla birlikte belirli yazılım işlevlerini sistematik ve hızlı bir şekilde kontrol edilmesine olanak tanımaktadır. Özellikle kapsamlı projelerde tercih edilen otomasyon testleri, test süreçlerinde zaman tasarrufu sağlamakta ve yazılım kalitesinin daha verimli bir biçimde güvence altına alınmasına imkân vermektedir. (Jeremic ve diğerleri, 2023; Singh ve diğerleri, 2014).

2.2.1. Test Otomasyon

Yazılım geliştirme süreçlerinde, ürünlerin güvenilirliğini ve işlevselliğini sağlamak amacıyla test faaliyetleri kritik bir rol oynamaktadır. Geleneksel test yöntemleri, manuel olarak yürütüldüğünde insan faktörü işin içine dahil olduğu için hem zaman alıcı hem de hata yapmaya açık durumlar ortaya çıkmaktadır. Yazılım projelerinin karmaşıklığının ve sürüm döngülerinin hızla artması, manuel testlerin yetersiz kalmasına yol açmış ve test süreçlerinin otomatikleştirilmesini zorunlu hale getirmiştir. Test otomasyonu, yazılım sistemlerinde olası hataları erken aşamada tespit etmek, kalite güvencesini sağlamak ve yazılım geliştirme süreçlerini hızlandırmak amacıyla çeşitli araçlar ve yazılım test çerçeveleri kullanılarak gerçekleştirilen sistematik bir yaklaşımdır. Otomasyon testleri, belirlenmiş test senaryolarının yazılım araçları aracılığıyla kesintisiz bir şekilde yürütülmesini sağlar. Otomasyon testleri, manuel testlerin zaman alıcı yapısını ortadan kaldırarak test süreçlerinin daha hızlı ve verimli ilerlemesine olanak tanır (Harmancı ve diğerleri, 2015).

Otomasyon testleri, belirli koşullar altında manuel testlere kıyasla daha verimli sonuçlar sunabilmektedir. Otomasyon testlerinin tercih edilmesinin uygun olduğu başlıca durumlar şunlardır (Yener ve diğerleri, 2019: 179).

- Yapılan testin sürekli tekrarlanması gerekliyse,
- Detaylı test sonuç raporu bekleniyorsa,
- Testin hızlı bir şekilde sonuçlanması hedefleniyorsa,

- İnsan kaynaklı hatalar istenmiyorsa,
- Hata bulma oranı arttırılmak isteniyorsa,

Test otomasyon denilince akla sadece test yapmak gelebilir fakat çoğu zaman bahsedilen işlem bir yazılım geliştirmedir. Genel olarak test otomasyonu, yazılımların doğruluğunu ve işlevselliğini denetlemek amacıyla, yine bir yazılım geliştirme aracılığıyla test sürecinin otomatize edilmesi şeklinde tanımlanmaktadır (Yiğit Akbaş, 2016: 36). Geleneksel olarak süreç, kodlama bilgisi gerektirse de günümüzde kullanıcıların kod yazmadan test senaryoları oluşturabilmesine imkân tanıyan platformlar da geliştirilmiştir. Tasarlanan sistemler, genellikle sürükle-bırak gibi görsel arayüzler üzerinden test adımlarının tanımlanmasına dayanmaktadır fakat kod gerektirmeyen (no-code) test otomasyon sistemleri, açık kaynaklı araçların aksine ücret karşılığında kullanıma izin vermektedir.

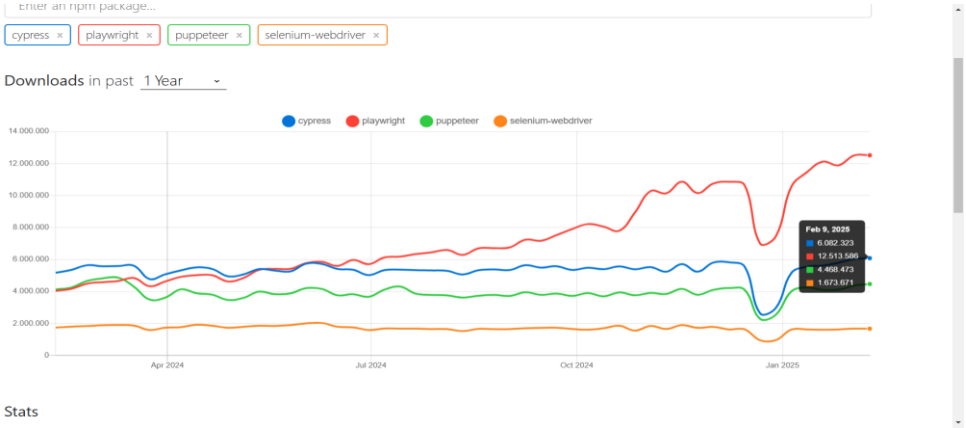
2.2.2. Test Otomasyon Araçları

Otomasyon testlerini yapabilmek için manuel testlerden farklı olarak bir araç kullanmak gerekmektedir. Yazılım test otomasyon araçları, yazılım uygulamalarının test süreçlerini insan müdahalesi olmadan çalıştırmaya yarayan yazılımlardır (browserstack, 2025). Literatürde genellikle piyasadaki test otomasyon araçlarının tam sayısına ilişkin net bir rakam bulunmamaktadır, çünkü sürekli yeni araçlar geliştirilmekte, mevcut araçlar güncellenmekte veya kullanımı sona ermektedir. Akademik araştırmalarda ve endüstri raporlarında çoğunlukla en popüler veya yaygın kullanılan araçların sayısına ve kategorilerine yer verilmektedir. Örnek olarak, 2023 yılında yapılan bir araştırmada test otomasyonu için kullanılan araçların sayısının yüzlerce olduğuna, ancak yaygın olarak kullanılanların sayısının yaklaşık 40-50 olduğuna dikkat çekilmektedir (Kertusha ve diğerleri, 2023).

Birçok test otomasyon aracı içerisinden doğru aracı seçmek oldukça zor ve önemli bir problemdir. Harmancı ve arkadaşlarının. 2025'te yaptığı çalışmaya göre doğru otomasyon aracı seçmenin otomasyon testi sürecinin başarılı olmasına direkt etki ettiğini gözlemlenmiştir. Bu bağlamda tez içerisinde dört popüler test otomasyonu aracı incelenecektir ve karşılaştırılmalı analizleri yapılacaktır. Söz konusu araçlar; Selenium, Cypress, Puppeteer ve Playwright olarak belirlenmiştir.

Tez kapsamında incelenmek üzere ele alınan dört farklı test otomasyonu aracının seçilmesinin temel nedeni, araçların özellikle web tabanlı yazılımların test edilmesi amacıyla geliştirilmiş olmaları, kullanıcılar tarafından en çok indirilen araçlar arasında yer almaları ve açık kaynaklı olmalarıdır. Araçların geliştirilmesinde katkı sağlayan topluluk desteği yüksekliği, dokümantasyon zenginliği ve web uygulamalarına yönelik test süreçlerinde sağladıkları çözümler ise araçların seçiminde destekleyici nedenlerdir.

Şekil 3: Araçların günlük indirme sayıları



Kaynak: npmtrends.com, 2022.

Yalnızca node paket yönetimi (Node Package Manager: NPM) verileri dikkate alındığında dahi, en düşük resmi indirilme sayısının günlük 1.673.671 ile Selenium aracı olduğu görülmektedir (9 Şubat 2025) (Şekil 3). Bununla birlikte, Selenium ve Playwright gibi araçlar, npm dışında farklı programlama dilleri ve paket yönetim sistemleri aracılığıyla da kullanılabildiğinden, araçların gerçek indirilme sayılarının tabloda belirtilenden çok daha yüksek olduğu değerlendirilmektedir.

2.2.2.1. Selenium

Selenium, 2004 yılında ThoughtWorks şirketinde Jason Huggins tarafından, firmanın dahili web tabanlı uygulamalarını test etmek amacıyla geliştirilmiştir. İlk olarak 'JavaScriptTestRunner' adıyla ortaya çıkan araç, süreç içerisinde geliştirilerek açık kaynaklı bir projeye dönüşmüştür. Huggins ve çalışma arkadaşlarının katkılarıyla Selenium Uzaktan Kontrol (Remote Control: RC) ve Selenium Grid gibi alt bileşenler geliştirilmiş; daha sonra Simon Stewart tarafından geliştirilen WebDriver projesi ile birleşerek Selenium 2 sürümü ortaya çıkmıştır. (Selenium, 2025).

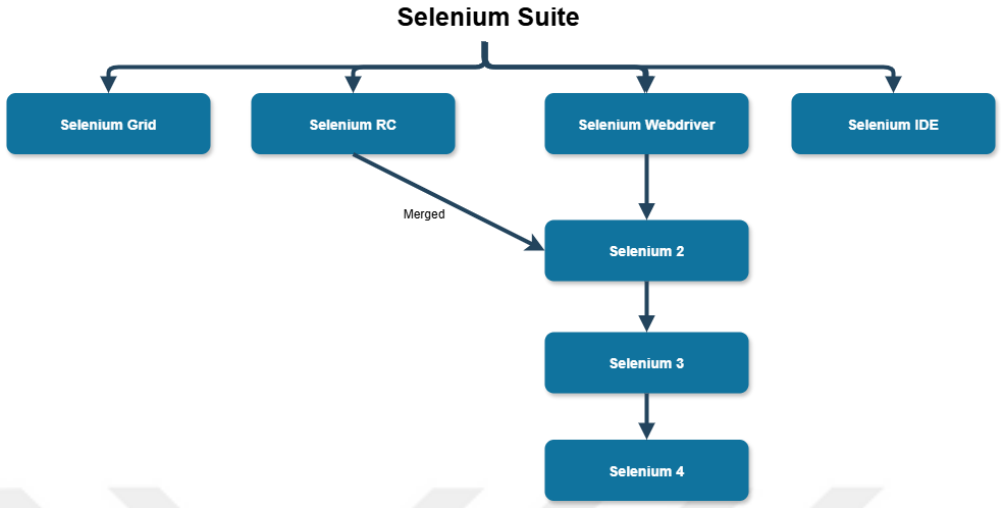
Selenium aracı günümüzde aktif olarak web uygulamalarını test etmek amacıyla kullanılmakta ve açık kaynaklı olarak geliştirilmeye devam etmektedir. Selenium, farklı ihtiyaçlara yönelik çeşitli bileşenler içeren bir araç setinden oluşmaktadır. Bileşenleri; Selenium RC, Selenium Grid, Selenium WebDriver ve Selenium Bütünleşik Geliştirme Ortamı (Integrated Development Environment: IDE) şeklinde sıralanabilir. Her bir bileşen, Selenium'un gelişim sürecinde farklı işlevler üstlenmiş ve zamanla test süreçlerinin daha esnek ve verimli yürütülmesine katkı sağlamıştır (lambdatest, 2023).

Tablo 3: Selenium Bileşenleri

Bileşen Adı	Açıklama
Selenium RC	Selenium Remote Control, tarayıcı otomasyonunu sağlamak için JavaScript kullanılarak geliştirilen ilk çözümdür. 2004 yılında geliştirilmiş olup, istemci-sunucu yapısı kullanır. Selenium WebDriver'ın ortaya çıkışıyla birlikte artık desteklenmemektedir.
Selenium Grid	Selenium Grid, aynı anda birden fazla tarayıcıda ve farklı makinelerde paralel testlerin yürütülmesini sağlayan bileşendir. Dağıtık test ortamlarının oluşturulmasına imkân tanır.
Selenium IDE	Selenium IDE, kullanıcıların test senaryolarını kod yazmadan, tarayıcı üzerinde kaydedip çalıştırmalarını sağlayan bir eklentidir. Özellikle test otomasyonuna yeni başlayanlar için kullanıcı dostu bir arayüz sunar. Google Chrome, Mozilla Firefox ve Microsoft Edge üzerinden kullanılabilir.
Selenium WebDriver	Selenium WebDriver, Selenium aracının en yaygın kullanılan bileşenlerinden biridir. Webdriver sayesinde otomasyon kodu ile tarayıcı etkileşim haline girebilir ve program DOM elementlerini manipüle edip kullanıcı davranışlarını taklit edebilir.

Kaynak: (lambdatest, 2023). Yazar tarafından derlenmiştir.

Şekil 4: Selenium Mimarisi



Kaynak: (edureka.co, 2025). Yazar tarafından derlenmiştir.

2.2.2.1.1. Selenium Özellikleri

Gelişim süreci boyunca farklı sürümler ve bileşenlerle zenginleştirilmiş olan Selenium, sağladığı esneklik ve geniş kullanım alanı ile yazılım test süreçlerinde önemli bir yer edinmiştir. Selenium'un öne çıkan başlıca özellikleri Tablo 4 içerisinde gözlemlenebilir.

Tablo 4: Selenium Özellikleri

Özellik	Açıklama
Çoklu Tarayıcı Desteği	Chrome, Firefox, Safari, Edge gibi yaygın tarayıcılarda çalışabilmektedir.
Çoklu Programlama Dili Desteği	Java, Python, C#, Ruby, JavaScript ve Kotlin gibi çeşitli programlama dilleri ile test senaryoları yazılabilmektedir.
Platform Bağımsız Çalışma	Windows, MacOS ve Linux gibi farklı işletim sistemlerinde kullanılabilmektedir.
Açık Kaynaklı ve Ücretsiz	Çeşitli topluluklar tarafından desteklenen ve sürekli geliştirilen açık kaynak kodlu bir projedir.
Paralel Test Çalıştırma	Selenium Grid ile aynı anda birden fazla tarayıcıda ve platformda paralel test çalıştırma imkânı sunar.
Diğer Araçlarla Entegrasyon	TestNG, JUnit gibi test çerçeveleri ile ve Jenkins gibi sürekli entegrasyon araçlarıyla kolay entegre edilebilmektedir.
No-Code Platform Desteği	Selenium IDE sayesinde kullanıcılara herhangi bir kod yazmadan otomasyon testi yatırabilme imkanı sağlar.
Dinamik Web Öğeleri ile Çalışabilme	AJAX, JavaScript gibi dinamik içeriklerle çalışan modern web uygulamalarını test edebilme yeteneğine sahiptir.

Kaynak: (selenium.dev, 2023). Yazar tarafından derlenmiştir.

2.2.2.1.2. Selenium Kullanım Alanları

Selenium, sahip olduğu geniş özellik yelpazesi ve sunduğu işlevsel yetkinlikler sayesinde, test otomasyonu süreçlerinde farklı ihtiyaçlara cevap verebilen çok yönlü bir araç haline gelmiştir. Aşağıda yer alan Tablo 5 içerisinde Selenium'un başlıca kullanım alanları özetlenmiştir.

Tablo 5: Selenium Kullanım Alanları

Yapılabilecek Testler/İşlemler	Açıklama
Uçtan Uca (End-to-End) Test	Web uygulamalarının kullanıcı arayüzünde yer alan işlevlerini belirli bir sırada ve beklenilen şekilde uçtan uca çalışıp çalışmadığının doğrulanmasında kullanılır. Basit anlamda kullanıcı davranışını simüle etmek için yapılan testtir.
Performans Testi	Web uygulamasının farklı kullanıcı yükleri veya farklı ortamlar altında nasıl çalıştığını incelemek için kullanılır. (Selenium ile performans testi yapmak best practice olarak önerilmez!)
API Testi	Selenium doğrudan api testini desteklemez fakat api testi için tasarlanmış özel kütüphaneler (Örneğin Rest Assured) ile uyumlu çalışarak api testlerini gerçekleştirebilir.
Sürekli Entegrasyon	Jenkins gibi sürekli entegrasyon araçları ile entegre edilerek, yazılım geliştirme sürecinde yapılan her güncellemenin ardından otomatik testlerin çalıştırılmasına imkân tanır.
Tarayıcı Otomasyonu ve Web Scraping (Veri Toplama):	Özellikle veri çekme (web scraping) gibi tarayıcı otomasyonuna dayalı işlemlerde, sayfalardan belirli bilgilerin otomatik olarak toplanması amacıyla kullanılabilmektedir.

Kaynak: (selenium.dev, 2023). Yazar tarafından derlenmiştir.

2.2.2.2. Cypress

Cypress, 2014 yılında Brian Mann tarafından, modern web uygulamalarının test süreçlerini kolaylaştırmak amacıyla geliştirilmiştir. Mann, o dönemde mevcut test araçlarının yetersizlikleri ve karmaşıklıkları nedeniyle, daha hızlı, kolay ve güvenilir bir test otomasyon aracı oluşturma ihtiyacı hissetmiştir. Mevcut ihtiyacı doğrultusunda, Cypress'i tarayıcı içinde çalışan ve gerçek zamanlı geri bildirim sağlayan bir araç olarak tasarlamıştır. 2018 yılının Kasım ayında resmi olarak piyasaya sürülen Cypress, kısa sürede geliştiriciler ve kalite güvence uzmanları arasında

popülerlik kazanarak günümüzde en yaygın kullanılan otomasyon araçları arasında yerini almıştır. (cypress.io, 2020).

Cypress, kullanıcıların ve işletmelerin gereksinimlerine yönelik olarak dört farklı ürün sunmaktadır. Ürünlerden üçü—Cypress Cloud, UI Coverage ve Cypress Accessibility—ücretli olarak piyasaya sürülmüş olup, gelişmiş test yönetimi, erişilebilirlik analizleri ve test kapsamı değerlendirme gibi ek özellikler sağlamaktadır. Bununla birlikte, Cypress'in en kapsamlı ve temel ürünü olan Cypress App, açık kaynak kodlu olarak geliştirilmeye devam etmekte ve kullanıcıların ücretsiz erişimine sunulmaktadır. Tablo 6 içerisinde ürünlerin kısa açıklamaları bulunmaktadır.

Tablo 6: Cypress Ürünleri

Ürün adı	Açıklaması	Ücretli mi?
Cypress App	Açık kaynak kodlu, web uygulamalarının uçtan uca test edilmesini sağlayan temel Cypress aracı.	Ücretsiz
UI Coverage	Kullanıcı arayüzü (UI) kapsamını analiz ederek hangi alanların test edildiğini görsel olarak raporlar.	Ücretli
Cypress Accessibility	Engelli insanların web uygulamasını kullanabilmesi için kalkması gereken engelleri tespit eder.	Ücretli
Cypress Cloud	Test sonuçlarını kaydetme, analiz etme, CI/C entegrasyonu ve paralel test çalıştırma desteği sunar.	Ücretli

Kaynak: Yazar tarafından derlenmiştir (docs.cypress.io,2020).

Cypress ürünlerinin kısaca ne olduğu, neler yapabildiği ve ücretli olup olmayışı tablo 6 üzerinde sunulmuştur.

2.2.2.2.1. Cypress App özellikleri

Cypress uygulamasının ücretsiz sunduğu ana ürünü içerisinde birçok farklı özellik bulunur. Uygulamanın sahip olduğu özellikler Tablo 7’de gözlemlenebilir:

Tablo 7: Cypress Uygulaması Özellikleri

Özellik Adı	Açıklama
Zaman Yönetimi	Test çalıştırma sırasında her adımın anlık görüntülerini alarak test sürecini görselleştirir ve hata ayıklamayı kolaylaştırır.
Hata analizi	Test başarısızlıklarının nedenlerini doğrudan geliştirici araçları üzerinden analiz etme imkânı sunar. Neden testin başarısız olduğunu öğrenmeyi kolaylaştırır.
Otomatik Bekleme	Test adımları arasına otomatik olarak bekleme süreleri koyar ve böylece asenkron kod çalıştırma hataları hakkında ekstra bir geliştirmeye gerek kalmaz.
Ağ trafik kontrolü	Ağ trafiğini kontrol etme ve durum testlerini sunucuya bağımlı olmadan çalıştırma imkânı sağlar.
Kararlı Sonuçlar	Selenium veya WebDriver kullanmadan doğrudan tarayıcı üzerinde çalışarak daha kararlı ve hızlı testler sunar.
Çoklu Tarayıcı Desteği	Chrome, Firefox, Edge ve Electron gibi tarayıcılarda test çalıştırabilir.

Kaynak: Yazar tarafından derlenmiştir (docs.cypress.io, 2020).

2.2.2.2.2. Cypress Cloud özellikleri

Cypress'in ücretli olan ve ekstra özelliklerini barındıran Cypress Cloud ürününün özellikleri Tablo 8 üzerinde detaylıca açıklanmıştır.

Tablo 8. Cypress Cloud Uygulamasının Özellikleri

Özellik Adı	Açıklama
Test Tekrarı	Çalıştırılan testlerin kaydedilmesini ve daha sonra tekrar oynatılmasını sağlayarak hata tespitini kolaylaştırır.
Akıllı Yönetim	Paralel test çalıştırma, başarısız testlerin öncelikli olarak yürütülmesi ve başarısız olan testlerin otomatik iptal edilmesi gibi özellikler sunar.
Kararsız (flaky) Test Tespiti	Kararsız testleri tespit ederek güvenilirliği artırmaya yönelik analizler sağlar.
Dal Analizi:	Kod değişikliklerinin test sonuçlarına etkisini tek bir ekran üzerinden analiz etme imkânı sunar. Kararsız ve başarısız testlerin olduğu kodları asıl kod ile birleştirmeden önce tespit eder.
Entegrasyon	GitHub, GitLab ve Bitbucket gibi platformlarla entegre çalışarak test sonuçlarını doğrudan sürüm kontrol sistemlerinde görüntüleme imkânı tanır.
Test Analizi	Test sonuçlarını zaman içinde analiz ederek geriye dönük hata ve iyileştirme trendlerini tespit etme olanağı sağlar.

Kaynak: Yazar tarafından derlenmiştir. (docs.cypress.io, 2020).

2.2.2.2.3. UI Coverage ve Cypress Accessbility Özellikleri

Cypress'in sunduğu ücretli araçlar olan fakat daha az özellik bulunduruan Ui Coverage ve Cypress Accessbility'nin özellikleri Tablo 9 içerisinde ortak biçimde açıklanmıştır.

Tablo 9: Ui Coverage ve Cypress Accessibility Ürünleri Özellikleri

Ürün Adı	Özellik Adı	Açıklama
UI Coverage	**Görsel Kapsama Desteği**	Web uygulaması üzerinde her bir sayfadaki test edilen alanları görselleştirerek herkesin anlayacağı şekilde görsel bir sonuç ortaya çıkarır. Sadece Yazılım Test mühendisi değil aynı zamanda son kullanıcıda hangi alanların test edildiğini görebilir ve anlayabilir.
	Result Api Kullanımı	Görsel kapsam sonuçlarını programatik olarak alıp mevcut test iş akışlarına entegre etme imkânı sunar. Ne kadar alanın test edildiği iş akışı içerisinde raporlanabilir hale gelir.
Cypress Accessbility	**Erişebilirlilik Kontrolü**	Cypress testlerine binlerce erişilebilirlik kontrolünü kolayca entegre eder ve kod değişikliği gerektirmeden erişilebilirlik hatalarını tespit eder.
	Erişebilirlik Raporları	Çalıştırılan testler için erişilebilirlik açısından detaylı raporlar oluşturur.

Kaynak: Yazar tarafından derlenmiştir (docs.cypress.io, 2020).

Cypress'in sunduğu Cypress Cloud, Ui Coverage ve Cypress Accessbility ürünlerinin ücretlendirmeleri isteyen kullanıcı ve şirketler tarafından Cypress'in resmi sitesinden ulaşılabilir (https://www.cypress.io/pricing).

2.2.2.2.4. Cypress Kullanım Alanları

Cypress, modern web uygulamalarının farklı test süreçlerini destekleyebilen kapsamlı bir test otomasyon aracıdır. Tarayıcı tabanlı testlerin yanı sıra, bileşen testleri, erişilebilirlik değerlendirmeleri ve API testleri gibi çeşitli alanlarda etkin bir şekilde kullanılmaktadır. Sağladığı esneklik ve genişletilebilir yapı sayesinde, geliştiriciler ve kalite güvence ekipleri tarafından yaygın olarak tercih edilmektedir. Cypress'in başlıca kullanım alanları Tablo 10'da listelenmiştir:

Tablo 10. Cypress Kullanım Alanları

Yapılabilecek Testler/işlemler	Açıklama
Uçtan Uca (End-to-End) Testler	Cypress'in temel kullanım alanlarından biri olup, web uygulamalarının kullanıcı arayüzü (UI) üzerinden tam kapsamlı test edilmesini sağlar. Tarayıcıda uygulamayı açarak, bir kullanıcının gerçekleştirdiği işlemleri simüle eder ve otomatikleştirir.
Bileşen (Component) Testleri	React, Angular, Vue ve Svelte gibi modern UI kütüphaneleri için geliştirilmiş bileşenleri izole bir ortamda test etmeye imkân tanır. Her bir bileşenin bağımsız olarak doğrulanmasını sağlayarak geliştirme sürecini hızlandırır.
Erişilebilirlik (Accessibility) Testleri	Cypress, çeşitli eklentilerle birlikte erişilebilirlik değerlendirmeleri yaparak, web uygulamalarının belirlenen standartlara uygun olup olmadığını test edebilir. Cypress Cloud ile entegre edildiğinde, erişilebilirlik ihlalleri hakkında detaylı raporlar sunar.
Kullanıcı Arayüzü Kapsamı (UI Coverage)	Uygulamanın test kapsamını analiz ederek eksik veya test edilmemiş bölümleri tespit eder. Bu sayede, kritik kullanıcı akışlarının gözden kaçırılmasının önüne geçilir ve yazılımın güvenilirliği artırılır.
API Testleri	Cypress, yalnızca tarayıcı tabanlı testlerle sınırlı kalmayıp, doğrudan HTTP istekleri göndererek API testlerini de gerçekleştirebilir.
Diğer Test Türleri ve Entegrasyonlar	Cypress, resmi ve üçüncü taraf eklentiler aracılığıyla test kapsamını genişletebilir. Özel HTTP çağrıları, performans testleri ve farklı CI/CD araçlarıyla entegrasyon gibi çeşitli test senaryoları desteklenmektedir.

Kaynak: Yazar tarafından derlenmiştir (docs.cypress.io, 2020).

2.2.2.3. Puppeteer

Puppeteer, modern web tarayıcılarını kontrol etmek ve otomatikleştirmek için kullanılan bir Node.js kütüphanesidir. Google tarafından geliştirilmiş olup, ilk sürümü Ocak 2018'de yayınlanmıştır. Puppeteer, Chrome veya Chromium tarayıcılarını başsız (headless) veya başlı (headful) modda çalıştırarak, web sayfaları üzerinde otomasyon işlemleri gerçekleştirmeye olanak tanır. Başsız mod sayesinde, geliştiriciler ve test uzmanları, tarayıcı tabanlı görevleri programatik olarak yönetebilirler (pptr.dev, 2023).

Puppeteer'ın geliştirilme amacı, web tarayıcılarının otomasyonunu daha erişilebilir ve etkili hale getirmektir. Önceden, kullanıcı akışlarını test etmek veya web sayfalarıyla etkileşimde bulunmak için ya başlı tarayıcı otomasyonu (örneğin, Selenium ile Firefox) ya da kendi JavaScript motoruna sahip başsız tarayıcılar kullanılıyordu. Puppeteer, Chromium tarayıcı motorunu kullanarak, geliştiricilere hem başlı hem de başsız tarayıcı seçenekleri sunar ve böylece hız ve güvenilirlik arasında bir denge sağlar. Başsız tarayıcı (headless) modu sayesinde, test çalıştırma süreci sırasında grafiksel bir tarayıcı arayüzü açılmadan işlemler gerçekleştirilebilir. Başsız tarayıcı modu, testlerin daha hızlı yürütülmesine katkı sağlarken, görsel arayüzün neden olabileceği yüklenme gecikmeleri ve bekleme süresiyle ilgili hataların önüne geçilmesini mümkün kılmaktadır (kasada.io, 2021).

2.2.2.3.1. Puppeteer Özellikleri

Puppeterr test otomasyon aracı bünyesinde diğer test otomasyon araçlarına benzer ve farklı olan birçok önemli yeteneği bulunmaktadır. Puppeteer aracının özellikleri Tablo 11 üzerinde listelenmiştir.

Tablo 11. Puppeteer Aracı Özellikleri

Özellik Adı	Açıklama
Node.js Tabanlı ve Açık Kaynaklıdır	Google tarafından geliştirilmiş olup, Chromium tabanlı tarayıcıların otomasyonunu sağlamak amacıyla kullanılan açık kaynak kodlu bir Node.js kütüphanesidir.
Başsız (Headless) ve Başlı (Headful) Mod Desteği	Başsız modda (headless) çalıştırılarak testlerin görsel arayüz yükü olmadan daha hızlı tamamlanmasını

	sağlarken, başlı modda (headful) çalıştırılarak test sürecinin manuel gözlemlenmesine olanak tanır.
Chrome ve Chromium Tabanlı Tarayıcılar İçin Otomasyon Sağlar	Puppeteer, Chrome DevTools Protokolü üzerinden Chrome ve Chromium tarayıcılarını otomatikleştirebilir. WebDriver BiDi desteği ile Firefox üzerinde de test çalıştırılabilir.
Ekran Görüntüsü ve PDF Oluşturma	Web sayfalarının belirli bölümlerinin veya tamamının ekran görüntüsünü (screenshot) alabilir ve sayfaları PDF formatında kaydedebilir.
Ağ Trafiğini İzleyebilir ve Manipüle Edebilir	HTTP istek ve yanıtlarını yakalayarak test süreçlerinde sunucu yanıtlarını değiştirme veya hata senaryolarını oluşturma imkânı sunar.
Performans Analizi Yapabilir	Sayfa yükleme süreleri, ağ trafiği ve bileşen yükleme süreçlerini ölçerek, uygulama performansının değerlendirilmesini sağlar.
Mobil Cihaz Emülatörü	Farklı ekran boyutları ve cihaz türleri için tarayıcı davranışlarını taklit ederek mobil uyumluluk testleri gerçekleştirebilir.
Çoklu Sayfa ve Sekme Desteği	Aynı anda birden fazla sayfa veya sekmede test senaryolarının çalıştırılmasına imkân tanır.
Test Çerçeveleriyle Entegrasyon	Jest, Mocha ve Cucumber gibi popüler test çerçeveleriyle entegre olarak daha kapsamlı test süreçlerinin oluşturulmasını sağlar.

Kaynak: Yazar tarafından derlenmiştir (pptr.dev, 2023)

2.2.2.3.2. Puppeteer Kullanım Alanları

Puppeteer aracı sahip olduğu özellikleriyle birlikte özellikle web otomasyonu alanında oldukça kullanışlı olan araçlardan biri haline gelmiştir. Puppeteer ile yapılabilecek testler ve diğer işlemler Tablo 12 üzerinde özetlenmiştir.

Tablo 12. Puppeteer Aracı Kullanım Alanları

Yapılabilecek Testler/işlemler	Açıklama
Uçtan Uca (End-to-End) Test Otomasyonu	Kullanıcı etkileşimlerini simüle ederek buton tıklama, form doldurma ve sayfa gezinme gibi işlemleri otomatikleştirebilir.
Veri Toplama (Web Scraping)	Dinamik içerik barındıran web sayfalarından veri çekmek için Puppeteer kullanılabilir. JavaScript ile oluşturulan içerikleri işleyerek, geleneksel yöntemlerle erişilemeyen verilerin toplanmasını sağlar.
Ekran Görüntüsü ve PDF Oluşturma	Belirli web sayfalarının ekran görüntülerini almak veya sayfaları PDF formatında kaydetmek için Puppeteer kullanılabilir.
Performans ve Yük Testleri	Sayfa yükleme süreleri, ağ trafiği ve kaynak kullanımı gibi metrikleri izleyerek, web uygulamalarının performansını değerlendirmek ve iyileştirmek amacıyla kullanılır.

SEO ve Meta Veri Doğrulama	Arama motoru optimizasyonu (SEO) çalışmalarında, sayfaların meta verilerini ve içerik yapısını kontrol etmek için kullanılabilir.
Erişilebilirlik Testleri	Web sayfalarının erişilebilirlik standartlarına uyumunu kontrol etmek amacıyla, sayfa yapısını ve etkileşimlerini analiz etmek için Puppeteer'dan faydalanılabilir
İçerik İzleme ve Güncelleme	Belirli web sayfalarındaki içerik değişikliklerini izlemek ve bu değişikliklere göre otomatik bildirimler veya güncellemeler yapmak için kullanılabilir.

Kaynak: Yazar tarafından derlenmiştir (pptr.dev, 2023)

2.2.2.4. Playwright

Playwright, Microsoft tarafından 31 Ocak 2020'de piyasaya sunulmuştur. Araç, çapraz tarayıcı testleri için birleşik bir API ihtiyacını karşılamak ve mevcut araçlardaki (örneğin, Puppeteer ve Selenium) sınırlamaları aşmak amacıyla tasarlanmıştır. Geliştirme ekibi, daha önce Google'da Puppeteer projesinde çalışmış mühendisleri de içermektedir. Playwright, otomatik beklemeler, çoklu tarayıcı desteği ve ağ müdahalesi gibi özellikleriyle modern web testleri için güçlü bir araç olarak öne çıkmıştır. Tanıtımından yana, aktif olarak geliştirilmeye devam etmekte ve web test topluluğunda hızlı bir şekilde benimsenmektedir (testomat.io, 2024).

2.2.2.4.1. Playwright Özellikleri

Playwright aracı da tıpkı daha önce açıklanan test otomasyon araçları gibi çeşitli farklı özelliklere sahiptir. Aracın özellikleri Tablo 13'te açıklanmıştır.

Tablo 13: Playwright Aracının Özellikleri

Özellik Adı	Açıklama
Çoklu Tarayıcı Desteği	Tek bir API aracılığıyla Chromium, Firefox ve WebKit tarayıcılarında test imkânı sunar.
Farklı Programlama Dilleriyle Uyum	JavaScript, Python, C# ve Java gibi dillerde test senaryoları yazılmasına olanak tanır.
Otomatik Beklemeler	Elementlerin yüklenmesi veya belirli koşulların gerçekleşmesi için otomatik olarak bekleyerek, manuel müdahale ihtiyacını ortadan kaldırır.
Ağ Trafiği Kontrolü	Ağ isteklerini yakalama ve manipüle etme yeteneğiyle, farklı senaryoların test edilmesini sağlar.
Paralel Test Yürütme	Testlerin farklı tarayıcı ve sayfalarda eşzamanlı olarak çalıştırılmasına imkân tanır.

Mobil Tarayıcı Desteği	Mobil cihazlardaki tarayıcıları simüle ederek, mobil uyumluluk testlerinin yapılmasına olanak tanır.
Çapraz Platform Desteği	Windows, macOS ve Linux işletim sistemlerinde sorunsuz çalışır.
Açık Kaynak ve Aktif Geliştirme	Apache License 2.0 ile lisanslanmış olup, aktif bir topluluk ve sürekli güncellemelerle desteklenmektedir.

Kaynak: Yazar tarafından derlenmiştir (playwright.dev, 2023)

2.2.2.4.2. Playwright Kullanım alanları

Sahip olduğu çeşitli özellikler ve güncel olarak geliştirilmesi playwright aracını tıpkı daha önce açıkladığımız test otomasyon araçları gibi farklı alanlarda kullanılabilir bir araç haline getirmiştir. Tablo 14 üzerinde Playwright ile yapılabilecek testler ve diğer işlemler gözlemlenebilir.

Tablo 14: Playwright Aracı Kullanım Alanları

Yapılabilecek Testler/işlemler	Açıklama
Uçtan Uca (End-to-End) Test Otomasyonu	Playwright, web uygulamalarının kullanıcı etkileşimlerini gerçekçi bir şekilde simüle ederek, uygulamanın tüm iş akışlarını baştan sona test etmeyi sağlar.
Mobil Cihaz Testleri	Mobil cihazların tarayıcılarını simüle ederek, uygulamaların mobil uyumluluğunu test etme imkânı sunar. Sahip olduğu simülasyon özelliği, mobil kullanıcı deneyimini optimize etmek için kritik öneme sahiptir.
Bileşen (Component) Testleri	Playwright, izole edilmiş bileşen testleri yaparak, her bir yazılım bileşenin bağımsız olarak doğru çalıştığını doğrulama imkânı sunar.
API Testleri	HTTP isteklerini yakalama ve manipüle etme yeteneği sayesinde, arka uç servislerin ve API'lerin doğru çalışıp çalışmadığını test etme olanağı sağlar.
Erişilebilirlik (Accessibility) Testleri	Web uygulamalarının engelli bireyler için erişilebilirlik standartlarına uygunluğunu değerlendirmek için kullanılır.

Kaynak: Yazar tarafından derlenmiştir (playwright.dev, 2023).

ÜÇÜNCÜ BÖLÜM
ÇOK KRİTERLİ KARAR VERME

3.1. ÇOK KRİTERLİ KARAR VERME TANIMI

Geçmişten günümüze kadar insanlık, kendisi ve başkaları adına her zaman geleceğe yönelik kararlar vermek durumda kalmıştır. Verilen kararlar hem insanların sorunlarını çözmüş hem de teknolojinin gelişimini hızlandırmıştır. Teknolojinin gelişimi ve değişimi ise paralelinde verilecek kararların zorluk derecesini ve karmaşıklığını da arttırmaktadır. Gelinen durumda ise her bir verilen karar başka bir kararı etkilemekte ve birbirine bağlı karmaşıklaşan bir ortam oluşturmaktadır. Bu bağlamda verilecek kararların önemi oldukça artmıştır çünkü her bir karar, kararı veren insana geleceğe yönelik fayda ya da maddi bir zarar yaratacaktır ve birçok durumda geri dönülemez bir ortam oluşturacaktır (Yaralıoğlu ve Karagöz, 2024).

Karar vermenin en yalın haliyle tanımı; alternatif seçeneklerin değerlendirilerek içlerinden en uygun olanının belirlenmesi sürecidir. İnsanlar için en uygun alternatifin belirlenmesi, birden fazla kriterin eş zamanlı olarak değerlendirilmesini gerektirmektedir. Kriterler ise kendi içlerinde farklı özelliklere ve yapılara sahiptir. Başka bir ifadeyle, bazı kriterler için (tam sayı ile ifade edilebilen bir kriter, örneğin maliyet) pozitif bir tam sayı değeri almak o kriteri iyi bir kriter yaparken başka bir kriter için bu durum kriterin kötü olduğunu gösterebilir (Yaralıoğlu ve Karagöz, 2024).

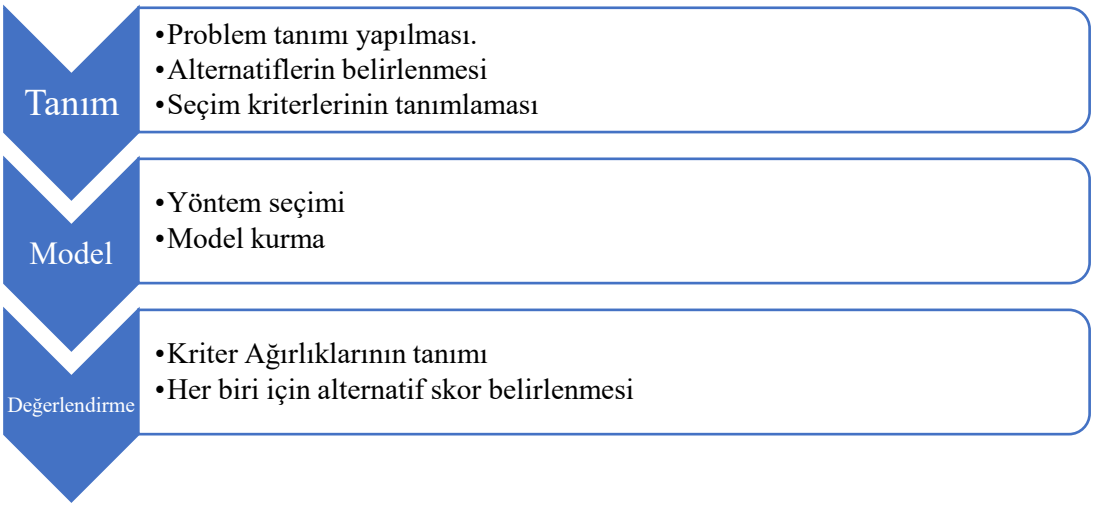
Günümüze bakıldığında özellikle şirketlerin bu tarz problemlerle karşılaştığını gözlemlenebilir. Örneğin bir işletme onu ayakta tutan kurumsal kaynak planlaması (Enterprise Resource Planning: ERP) sistemini seçmesi gerektiğinde kendi kurumuna özgü ihtiyaçlarını ve sistemi temin edeceği tedarikçiyi gibi birçok faktörü aynı anda değerlendirmesi gerekmektedir. Her ERP sistemi farklı özelliklere sahip olduğundan, işletme gereksinimlerine en uygun olanı seçmek için tüm kriterlerin dikkatli bir şekilde analiz edilmesini gerektirir. Yanlış bir seçim, operasyonel süreçlerde verimsizliklere ve mali kayıplara yol açabilir. (Aşan ve Aycin, 2020)

Mevcut durumda birçok firma ERP sistemi seçimine benzer ve fazlaca kritere sahip karmaşık karar verme problemleriyle karşılaşmaktadır. Problemlerin çözümü

için ise çok kriterli karar verme yöntemlerine başvurabilirler. Çok kriterli karar verme, birden fazla alternatifin ve genellikle birbirleriyle çelişen kriterlerin bulunduğu karar problemlerinde, sistematik ve analitik bir yaklaşım sunan yöntemler bütününe verilen isimdir. Yöntemler, karar vericilerin alternatifleri belirli bir ölçek kullanarak sıralamasına ve ilgili tüm faktörleri dikkate alarak en iyi seçimi yapmasına olanak tanır (Kılıç ve Asal, 2024).

Çok Kriterli Karar Verme Yöntemleri, nicel ve nitel seçim kriterlerini dikkate alarak en uygun alternatifin seçilmesini amaçlamaktadır. Bu yöntemler genellikle alternatifler, kriterler ve kriter ağırlıkları olmak üzere üç temel bileşenden oluşur. ÇKKV'nin en büyük avantajı, birden fazla kriteri ve alternatifi eş zamanlı olarak değerlendirebilmesidir. ELECTRE, TOPSIS, MARCOS, VIKOR, MAUT, COPRAS, UTA, SWARA, PROMETHEE ve MOORA gibi çeşitli ÇKKV yöntemleri, bireysel karar süreçlerinden işletmelerin stratejik ve kritik kararlarına kadar geniş bir kullanım alanına sahiptir. Literatürde, farklı sektörlerde faaliyet gösteren firmaların performans analizlerinde bu yöntemlerin sıklıkla kullanıldığı görülmektedir (Özcan ve Ömürbek, 2020: 79).

Şekil 5: ÇKKV Süreci



Kaynak: Yazar tarafından Türkçeleştirilmiştir. (Belton, V. ve Stewart, T.J., 2002)

Çok kriterli karar verme yöntemleri literatürde yaygın olarak objektif ve sübjektif olmak üzere iki ana gruba ayrılır. Yapılan ayrım, kriter ağırlıklarının belirlenme şekline ve yöntemin bilgi kaynağına dayanmaktadır. Objektif yöntemler,

karar vericinin doğrudan yargılarına ihtiyaç duymadan matematiksel modellerle veriye dayalı hesaplamalar yaparak kriter ağırlıklarını veya tercih sıralamasını belirleyen tekniklerdir. Buna karşılık sübjektif yöntemler, uzmanların veya karar vericilerin tercihlerine, tecrübelerine ve değerlendirmelerine dayanarak sonuç üreten, dolayısıyla insan yargısını sürece katan tekniklerdir. İki farklı yöntem grubunun bir arada kullanımıyla Hybrid (karma) yöntemler kullanılabilir (Keshavarz-Ghorabaee ve diğerleri, 2021). Çalışmanın yöntemi olarak geliştirilmiş bulanık SWARA ve bulanık MARCOS birleşimi olan karma yöntem kullanılacaktır. Kriter ağrılıkları geliştirilmiş bulanık SWARA ile belirlenecek ve alternatif araçların sıralamasını kriter ağırlıklarına dayandırılarak Bulanık MARCOS yardımıyla belirlenecektir.

Tablo 15: Objektif Ve Sübjektif Yöntemler

Kategorizasyon	Objektif Yöntemler (Veri Tabanlı)	Sübjektif Yöntemler (Uzman Tabanlı)
Bilgi Kaynağı	**Ham veriler** (istatistiksel ölçüler, bilgi teorisi veya optimizasyon modelleri uygulanır. Karar vericiden ön bilgi gerekmez)	**Uzman görüşü** (Karar vericiler kriterleri karşılaştırır, puanlar veya sıralar.)
Avantajlar	• Tarafsız ve tekrarlanabilir sonuç. • Yanlılık riski yok.	• Karar vericinin deneyimini yansıtır. • Uzman bilgisi ile daha gerçekçi sonuçlar üretir. • Kullanımı ve anlaşılması genelde kolaydır.
Dezavantajlar	• Karar vericinin deneyimini yansıtmaz. • Verideki ölçek etkileri sonucu etkileyebilir.	• Özneldir, uzmana göre sonuç değişebilir. • Kişisel önyargılar sonucu etkileyebilir.
Örnek Yöntemler	Entropi, CRITIC, MEREC, IDOCRIW, CILOS	ANP, AHP, SWARA, FUCOM, DEMATEL, MACBETH, SMART

Kaynak: Yazar tarafından derlenmiştir (Yacan, 2024; Keshavarz-Ghorabaee ve diğerleri, 2021).

3.2. BULANIK MANTIK VE ÇKKV İLİŞKİSİ

Klasik ÇKKV genellikle kesin verilerle işlem yapmayı gerektirirken, gerçek dünyadaki karar süreçleri belirsizlik, öznel yargılar ve bilgi eksiklikleri içermektedir. Bu nedenle, bulanık mantık tabanlı yaklaşımlar, karar vericilerin

değerlendirmelerindeki belirsizlikleri daha etkin yönetmek amacıyla ÇKKV yöntemleriyle entegre edilmiştir (Zavadskas ve Podvezko, 2016).

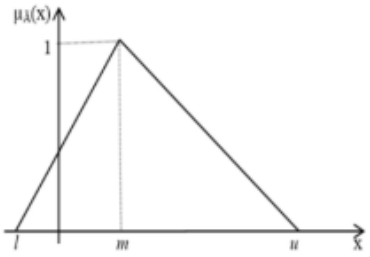
Bulanık mantık, karar verme süreçlerinde kesin doğruluk yerine, olasılık ve belirsizlik içeren yargıları modele dahil etmeyi sağlayan bir yaklaşımdır. Bulanık mantık çerçevesinde, bir elemanın belirli bir kritere ait olup olmadığı kesin olarak tanımlanmaz, bunun yerine belirli bir üyelik derecesi atanarak ele alınır. Örneğin, karar vericiler bir kriterin önem derecesini "yüksek", "orta" veya "düşük" gibi dilsel değişkenlerle ifade ettiğinde, bu ifadeler bulanık sayılar kullanılarak bulanık kümeler içerisinde modellenebilir (Keshavarz-Ghorabaee ve diğerleri, 2021).

3.2.1. Bulanık Küme

Bulanık küme teorisi, bulanık mantığın temel yapı taşlarından biridir ve karar süreçlerindeki belirsizlikleri yönetmek için kullanılır. Bilinen ve klasik olan küme teorisi, herhangi bir küme elemanını kümeye ait (1) veya ait değil (0) olarak kabul eder. Lotfi A. Zadeh isimli bilim insanı, bulanık küme terimini 1965'de literatüre kazandırmıştır, rastgele bir küme elemanının kümeye aitliğini [0,1] arasında bir değer alacak şekilde tanımlar ve bu sayede klasik küme anlayışını genişletir (Şengül ve Çağıl, 2020). Bu bağlamda, bulanık küme teorisi, ÇKKV yöntemlerine belirsizlikleri daha iyi modelleme ve değerlendirme olanağı sağlayarak karar verme sürecini daha esnek ve gerçekçi hale getirmektedir.

Normal ve dışbükey yapıya sahip olan bulanık kümeler, bulanık sayı olarak adlandırılmakta olup, karar analizlerinde işlemleri daha pratik hale getirmek amacıyla kullanılmaktadır. Bulanık kümelerin bir alt kümesi olarak tanımlanabilen bulanık sayılar, 4'e yakın, neredeyse 8 vb. kesinlik belirtmeyen sayılar üzerinde tanımlama yapılırken kullanılması oldukça verimlidir (Özçakar & Demir, 2011). Uygulamalarda genellikle tercih edilen bulanık sayılar üçgen veya yamuktur. Ã= (l, m, u) biçiminde gösterilen bulanık sayıya üçgen bulanık sayı denir. l, m ve u değerleri sıralı olarak minimum değer, en yüksek olasılığa sahip değer ve maksimum değer olarak ifade edilen gerçek sayı değerlerini alır. Bulanık sayısı şekil 2 üzerinde gösterilmektedir. (Şengül & Çağıl, 2020).

Şekil 6: Üçgen Bulanık Sayı



Kaynak: (Şengül & Çağıl, 2020).

Ã ‘nın kümeye aitlik fonksiyonu ise Şekil 3‘deki gibidir:

Şekil 7. Kümeye aitlik fonksiyonu

$$\mu_{\tilde{A}}(x;l,m,u) = \begin{cases} l \le x \le m & ise \quad \dfrac{(x-l)}{(m-l)} \\ m \le x \le u & ise \quad \dfrac{(u-x)}{(u-m)} \\ x > u \; veya \; x < l & ise \quad 0 \end{cases}$$

Kaynak: (Şengül & Çağıl, 2020).

3.3. SWARA METHODU

Kriter ağırlıklarının belirlenmesinde başvurulan ÇKKV yöntemlerinden biri olan Adım Adım Ağırlık Değerlendirme Oranı Analizi (Step-Wise Weight Assessment Ratio Analysis: SWARA), Kersuliene ve arkadaşları (2010) tarafından geliştirilmiştir. SWARA yönteminin literatürde yaygın biçimde kullanılmasının en önemli nedeni, diğer ağırlık hesaplama teknikleriyle karşılaştırıldığında uygulama açısından daha yalın bir yapıya sahip olmasıdır. Kriter ağırlıklarını belirleme sürecindeki adım sayısının AHP gibi yöntemlere göre daha az olması da SWARA’nın tercih edilmesinde etkili olmaktadır. Literatüre bakıldığında yöntemin, “uzman odaklı yöntem” olarak belirtildiğini görmek mümkündür. (Kara ve diğerleri, 2023)

SWARA yöntemi, karar vericilerin belirli bir kriterin diğer kriterlere göre önemini belirlemeleri için adım adım bir süreç izler. İlk olarak, karar vericiler her bir kriter için bir sıralama yapar ve ardından bu sıralamaya göre ağırlıklar belirlenir. Bu süreç, karar vericilerin kendi görüşlerini ve deneyimlerini yansıtmalarına olanak tanır

ve böylece daha özelleştirilmiş bir değerlendirme süreci ortaya çıkar (Aytekin ve Gündoğdu, 2021).

3.3.1. Geliştirilmiş Bulanık SWARA Yöntemi

Karar verme süreçlerinde kriter ağırlıklarının belirlenmesi, özellikle belirsizlik ve öznel yargıların söz konusu olduğu durumlarda büyük önem taşımaktadır. SWARA yönteminin, karar vericilerin subjektif değerlendirmelerini sistematik bir şekilde ele alarak kriterlerin ağırlıklandırılmasını sağlayan etkin bir yöntem olduğu önceki bölümlerde belirtildi ancak gerçek hayattaki karar verme problemleri genellikle belirsizlik içermekte olup, klasik SWARA yöntemi bu tür belirsizlikleri tam olarak yansıtmakta yetersiz kalmaktadır. Yetersizlik durumunu aşmak amacıyla geliştirilen geliştirilmiş bulanık SWARA, bulanık küme teorisinin SWARA yöntemi ile bütünleştirilmesi yoluyla kriter ağırlıklandırma sürecinde daha esnek ve güvenilir bir yapı sunmaktadır.

Yöntemin temel avantajı, karar vericilerin değerlendirmelerini Pythagorean bulanık kümeler (PFS) gibi gelişmiş bulanık sistemlerle modelleyerek, kriterlerin önem derecelerinin daha hassas bir şekilde belirlenmesine olanak tanımasıdır (He ve diğerleri, 2021). Böylece hem sübjektif yargıların tutarlılığı artmakta hem de karar verme süreçlerinde belirsizliklerin daha etkin bir şekilde yönetilmesi sağlanmaktadır.

Geliştirilmiş bulanık SWARA yöntemi dört adımdan oluşan belirli bir matematiksel işlemler bütününü takip eder (Zolfani ve diğerleri, 2021):

Formül 1: Geliştirilmiş Bulanık Swara Yöntemi Adımları

Adım 1. Kriterlerin Önem Derecelerinin Belirlenmesi: Geliştirilmiş Bulanık SWARA tekniğinin uygulanmasının ilk aşamasında, karar vericiler değerlendirme kapsamındaki kriterlerin önem sıralamasını belirlerler. Yapılan sıralamada, karar vericilerin görüşleri ve değerlendirmeleri doğrultusunda en yüksek öneme sahip kriter ilk sırada yer alırken, en düşük öneme sahip kriter ise son sıraya yerleştirilir.

Adım 2. Kriterler Arasında İkili Karşılaştırmaların Yapılması: Karar vericiler, kriterlerin birbirlerine göre önem derecelerini belirlemek amacıyla Tablo 2'de verilen dilsel ölçeği kullanarak karşılaştırmalar yaparlar. Her bir kriter (j), bir önceki kritere (j-1) göre göreceli olarak değerlendirilir. Değerlendirmeler sonucunda, kriterler

arasındaki önem oranları olarak sembolize edilir ve bu değerler kriterlerin birbirlerine göre ortalama karşılaştırmalı önem düzeylerini ifade eder.

Tablo 15: Dilsel Değişkenler Ölçeği

Dilsel Değişkenler	**Bulanık Sayılar**
Kesin bir şekilde daha az önemli (ALS)	(1.0, 1.0, 1.0)
Yüksek miktarda daha az önemli (DLS)	(0.50, 0.67, 1)
Çok daha az önemli (MLS)	(0.40, 0.50, 0.67)
Gerçekten daha az önemli (RLS)	(0.33, 0.40, 0.50)
Daha az önemli (LS)	(0.29, 0.33, 0.40)
Orta derecede daha az önemli (MDLS)	(0.25, 0.29, 0.33)
Zayıf derecede daha az önemli (WLS)	(0.22, 0.25, 0.29)
Eşit derecede önemli (ES)	(0, 0, 0)

Kaynak: Vrtagic et al., 2021 alınan kaynaktan yazar tarafından Türkçeleştirilmiştir.

Adım 3. Katsayı Değerinin Hesaplanması: Üçüncü aşamada kriterlere ait ağırlık değerleri yeniden hesaplanarak seçim kriterlerinin nihai göreceli önem düzeyleri belirlenir. Katsayı değerlerinin hesaplanması aşağıda yer alan 1, 2 ve 3 numaralı formüller yardımıyla hesaplanır:

$$\widetilde{k_j} = \begin{cases} \tilde{1}, & j = 1 \\ \widetilde{s_j} + 1, & j > 1 \end{cases} \qquad (1)$$

Ağırlık değerleri ($\widetilde{q_j}$) 2 numaralı denklem ile hesaplanır

$$\widetilde{q_j} = \begin{cases} \widetilde{1}, & j = 1 \\ \frac{\tilde{q}_{j-1}}{k_j}, & j > 1 \end{cases} \qquad (2)$$

Son olarak, kriterlere ilişkin bulanık ağırlık katsayıları aşağıda verilen Denklem (3) yardımıyla hesaplanmaktadır:

$$\widetilde{w_j} = \frac{\widetilde{q_j}}{\sum_{j=1}^{n} \widetilde{q_j}} \qquad (3)$$

$\widetilde{k_j}$: Kriterlerin katsayı değeri, $\widetilde{q_j}$: j'inci kriterin yeniden hesaplanmış bulanık ağırlığı, $\widetilde{w_j}$: j'inci kriterin bulanık göreceli ağırlık değerini ve n ise toplam kriter sayısını ifade etmektedir.

Adım 4. Kriter Ağırlıklarının Durulaştırılması: IMF SWARA yönteminin son adımında, bulanık değerlerin durulaştırılması işlemi aşağıda verilen Denklem (4) yardımıyla gerçekleştirilir:

$$w_{\text{net değer}} = \frac{w^{(l)} + 4w^{m} + w^{(u)}}{6} \qquad (4)$$

Literatürde, SWARA ve geliştirilmiş bulanık SWARA yöntemleri farklı alanlarda uygulanmış ve çeşitli çok kriterli karar verme problemlerinde etkinliği test edilmiştir. Aşağıdaki tabloda, SWARA ve geliştirilmiş bulanık SWARA yöntemleri üzerine yapılan akademik çalışmaların içeriği özetlenmiştir.

Tablo 16: SWARA ve Geliştirilmiş Bulanık SWARA İle Yapılan Çalışmalar

	Yazar/Yazarlar	Yıl	Yöntem	Konu
1	Kersuliene ve arkadaşları	2010	SWARA	SWARA yönteminin tanıtılması ve kriter ağırlıklandırmada kullanımı.
2	Zavadskas ve Podvezko	2016	SWARA	ÇKKV yöntemlerinde SWARA'nın nesnel kriter ağırlıklandırmadaki rolü.
3	Mardani ve arkadaşları	2017	SWARA	Ulaşım sistemleri için MCDM tekniklerinin incelenmesi ve SWARA'nın rolü.
4	Keshavarz-Ghorabaee ve arkadaşları	2018	Geliştirilmiş SWARA	SWARA yönteminin bulanık mantık ile genişletilmesi ve belirsiz karar ortamlarında uygulanması.
	He ve arkadaşları	2021	Geliştirilmiş Bulanık SWARA	Turizm sektöründ sürdürülebilirlik kriterlerinin belirlenmesi içi Pythagorean bulanı SWARA yöntem kullanımı.
6	Aytekin ve Gündoğdu	2021	SWARA-TOPSIS-SORT-B-WASPAS	OECD ve AB üyesi ülkelerin sürdürülebilir yönetişim düzeylerinin değerlendirilmesi.

7	Saeidi ve arkadaşları	2022	Geliştirilmiş Bulanık SWARA-TOPSİS	Üretim sektöründe sürdürülebilir insan kaynakları yönetimi için Pythagorean bulanık SWARA-TOPSIS yöntemi kullanımı.
8	Türkmen ve Demirel	2022	SWARA-Bulanık COPRAS	Biyogaz üretim sektöründe tedarikçi seçim sürecinin analiz edilmesi.
9	Erdem ve arkadaşları	2023	SWARA-Gri İlişkisel Analiz	Türkiye'deki dijital platform tercihlerinin belirlenmesi ve kriterlere göre sıralanması.
10	Kara ve arkadaşları	2023	SWARA-WASPAS	Süper Lig futbol kulüplerinin web sitelerinin performansa göre sıralanması.

Kaynak: Yazar tarafından derlenmiştir.

SWARA ve geliştirilmiş bulanık SWARA yöntemleri üzerine yapılan akademik çalışmalar, farklı sektörlerde karar verme süreçlerinin iyileştirilmesine yönelik çeşitli uygulamalar içermektedir. Kersuliene vd. (2010), SWARA yöntemini tanıtarak kriter ağırlıklandırmada nasıl kullanılacağını açıklamış, Zavadskas ve Podvezko (2016) ise SWARA'nın çok kriterli karar verme süreçlerinde nesnel kriter ağırlıklandırmada nasıl bir rol oynadığını incelemiştir. Keshavarz-Ghorabaee vd. (2018), SWARA yönteminin bulanık mantık ile genişletilerek belirsiz karar ortamlarında nasıl uygulanabileceğini araştırmış, Mardani vd. (2017) ise ulaşım sistemlerinde kullanılan ÇKKV tekniklerini inceleyerek SWARA yönteminin bu bağlamdaki önemini vurgulamıştır. He vd. (2021), turizm sektöründe sürdürülebilirlik kriterlerini belirlemek için Pythagorean bulanık SWARA yöntemini kullanırken, Saeidi vd. (2022) üretim sektöründe sürdürülebilir insan kaynakları yönetimi için Pythagorean bulanık SWARA-TOPSIS yönteminin etkinliğini değerlendirmiştir. Aytekin ve Gündoğdu (2021), Ekonomik kalkınma ve iş birliği ve Avrupa Birliği üyesi ülkelerin sürdürülebilir yönetişim düzeylerini belirlemek için WASPAS-TOPSIS-SWARA-SORT-B yöntemlerini kullanmış, Erdem vd. (2023) Türkiye'de faaliyet gösteren dijital platformların tercih edilme kriterlerini SWARA ve Gri İlişkisel Analiz ile incelemiştir. Kara vd. (2023), Süper Lig futbol kulüplerinin web sitelerinin

performanslarını değerlendirmek için SWARA ve WASPAS yöntemlerini kullanırken, Türkmen ve Demirel (2022) biyogaz üretim sektöründe tedarikçi seçim sürecini SWARA-Bulanık COPRAS yöntemiyle analiz etmiştir. İncelenen çalışmalar, SWARA ve geliştirilmiş bulanık SWARA yöntemlerinin farklı sektörlerde belirsizlik içeren karar süreçlerinde başarılı sonuçlar ürettiğini ortaya koymaktadır.

3.4. MARCOS Methodu

ÇKKV yöntemlerinden biri olan alternatiflerin ölçülmesi ve uzlaşma çözümüne göre sıralanması (Measurement of Alternatives and Ranking according to Compromise **S**olution: MARCOS) yöntemi, çok kriterli karar verme (ÇKKV) alanında görece yeni bir teknik olup 2020 yılında Stević ve çalışma arkadaşları tarafından literatüre kazandırılmıştır (Ecer ve Pamucar, 2021). Bu yöntem, karar problemlerinde ideal en iyi ve anti-ideal en kötü referans noktalarını dikkate alarak alternatiflerin bir "uzlaşık çözüme" göre değerlendirilmesini sağlar (Duc Trung, 2022).

MARCOS yönteminin literatürde hızlı kabul görmesinin ardında, diğer ÇKKV tekniklerine kıyasla sunduğu bazı avantajlar bulunmaktadır. Stević ve arkadaşları, MARCOS'un geleneksel yöntemlere (ör. MABAC, ARAS, WASPAS, TOPSIS gibi) göre üstün yönlerini ilk çalışmalarında ortaya koymuşlardır (Ecer ve Pamucar, 2021). Yapılan vaka analizlerinde, MARCOS ile elde edilen sıralamaların güvenilir ve tutarlı olduğu görülmüştür. Örneğin, El-Araby ve ekibi bir lokasyon seçimi probleminde MARCOS yöntemini diğer yaygın yöntemlerle karşılaştırmış ve MARCOS sonuçlarının diğer tekniklerle yüksek oranda korelasyon gösterdiğini**,** ayrıca ağırlık değişimlerine karşı sıralamanın büyük ölçüde istikrarlı kaldığını rapor etmişlerdir. Sonuç olarak MARCOS, kriter ağırlıklarındaki değişimlere veya farklı yöntemlerin kullanımına rağmen en iyi alternatifi tutarlı biçimde belirleyebilmiştir (El-Araby ve diğerleri, 2024).

3.4.1. Bulanık Marcos Yöntemi

Bulanık MARCOS yöntemi, MARCOS'un bulanık mantık ortamında uygulanmasıyla geliştirilmiş bir ÇKKV yaklaşımıdır. Gerçek dünyadaki karar

problemleri genellikle belirsizlik ve dilsel değerlendirmeler içerir; klasik MARCOS yöntemi ise karar vericilerin değerlendirmelerindeki bulanıklığı ve belirsizliği doğrudan ifade edememektedir. Bu eksikliği gidermek amacıyla MARCOS yöntemi bulanık küme teorisi ile genişletilmiştir. Bulanık MARCOS, uzman görüşlerindeki muğlaklık, belirsizlik ve eksik bilgileri modele dahil ederek daha esnek bir karar analizi sunar. Özellikle üçgensel bulanık sayılar gibi temsilimler kullanılarak, alternatiflerin ve kriterlerin değerleri bir aralık belirsizliğiyle ifade edilebilir. Böylece MARCOS'un ideal ve anti-ideal çözüm yaklaşımı, bulanık ortamda da karar vericilerin sezgisel yargılarını yansıtabilecek şekilde uyarlanır. Sonuç olarak, bulanık MARCOS yöntemi sayesinde MARCOS'un güçlü yönleri korunurken, karar vericilerin belirsiz tercihleri modele entegre edilerek daha gerçekçi ve güvenilir analizler yapılabilir (Ecer ve Pamucar, 2021).

Bulanık MARCOS yöntemi de diğer ÇKKV yöntemleri gibi belirli matematiksel işlemlerin ardı ardına tekrarlanarak uygulanmasıyla gerçekleştirilir. On farklı adımı içeren yöntem Formül 2'de belirtilmiştir (Stankovic ve diğerleri, 2020).

Formül 2: Bulanık MARCOS Yöntemi Adımları

Adım 1: Başlangıç karar verme matrisinin oluşturulması

Karar vericiler her bir araç için belirlenmiş kriterleri tablo 3 üzerindeki sözel ifadeler ile değerlendirirler ve yapılan değerlendirmeler sonucunda başlangıç karar verme matrisi oluşturulur.

Tablo 17: MARCOS Dilsel Değişkenler Ölçeği

Sözel İfade	l	m	u
Çok Kötü (ÇK)	1	1	2
Kötü (K)	1	2	3
Orta Kötü (OK)	2	3	4
Orta (O)	3	4	5
Orta İyi (Oİ)	4	5	6
İyi (İ)	5	6	7
Çok İyi (Çİ)	6	7	7

Kaynak: (Özdadoğlu ve diğerleri, 2021: 288)

L: Üçgen bulanık sayı için alt limit değeri

M: Üçgen bulanık sayı için orta nokta değeri.

U: Üçgen bulanık sayı için üst limit değeri.

Adım 2: Bulanık Genişletilmiş karar matrisi oluşturulması

Matrisin oluşturulması için en ideal olan (AI) ve en ideal olmayan (AAI) değerler kriterlere göre denklem 5 ve 6 ile belirlenir.

$$X = \begin{array}{c} \\ AAI \\ A_1 \\ A_2 \\ \cdots \\ A_m \\ AI \end{array} \begin{array}{c} \begin{array}{cccc} C1 & C2 & \ldots. & C3 \end{array} \\ \begin{bmatrix} x_{aa1} & x_{aa2} & \cdots & x_{aan} \\ x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & .. \\ x_{m1} & x_{22} & \cdots & x_{mn} \\ x_{ai1} & x_{ai2} & \cdots & x_{ain} \end{bmatrix} \end{array}$$

$$\tilde{A}(AI) = \min_i \tilde{x}_{ij} \text{ if } j \in B \text{ ve } \max_i \tilde{x}_{ij} \text{ if } j \in C \quad (5)$$

$$\tilde{A}(\mathrm{ID}) = \max_i \tilde{x}_{ij} \text{ if } j \in B \text{ ve } \min_i \tilde{x}_{ij} \text{ if } j \in C \quad (6)$$

B maksimizasyon grup kriterlerine aitken C ise minimizasyon grubu kriterlerine aittir.

Adım 3: Normalize karar matrisinin oluşturulması

Normalize karar matrisini $\widetilde{N} = [\tilde{n}_{ij}]_{\mathrm{mxn}}$ oluşturmak için gereken değerler 7 ve 8 numaralı denklemlere göre belirlenir.

$$\tilde{\mathrm{n}}_{ij} = \left(n_{ij}^l, n_{ij}^m, n_{ij}^u\right) = \left(\frac{x_{id}^l}{x_{ij}^u}, \frac{x_{id}^l}{x_{ij}^m}, \frac{x_{id}^l}{x_{ij}^l}\right), \quad \text{if } j \in C \quad (7)$$

$$\tilde{\mathrm{n}}_{ij} = \left(n_{ij}^l, n_{ij}^m, n_{ij}^u\right) = \left(\frac{x_{ij}^l}{x_{id}^u}, \frac{x_{ij}^m}{x_{id}^u}, \frac{x_{ij}^u}{x_{id}^u}\right), \quad \text{if } j \in B \quad (8)$$

x_{ij}^l , x_{ij}^m, x_{ij}^u ve x_{id}^m , x_{id}^l , x_{id}^u değerleri $\widetilde{X}$ matrisindeki değerleri temsil eder.

Adım 4: Ağırlıklandırılmış karar matrisinin oluşturulması

Ağırlıklandırılmış karar matrisinin $\widetilde{\mathrm{V}} = \left[\tilde{v}_{ij}\right]_{\mathrm{mxn}}$ hesaplanması matrix $\widetilde{N}$ ile kriterlerin bulanık ağırlık katsayılarının ($\widetilde{w}_j$) çarpılmasıyla oluşturulur (Denklem 9).

$$\tilde{v}_{ij} = \left(v_{ij}^l, v_{ij}^m, v_{ij}^u\right) = \tilde{n}_{ij} \otimes \widetilde{w}_i = \left(n_{ij}^l \times w_j^l, n_{ij}^m \times w_j^m, n_{ij}^u \times w_j^u\right) \quad (9)$$

Adım 5: Bulanık karar matrisinin oluşturulması

Bulanık karar matrisi denklem 10 ile hesaplanır.

$$\tilde{\mathrm{S}}_{\mathrm{i}} = \sum_{i=1}^{n} \tilde{v}_{ij} \quad (10)$$

$\tilde{S}_i(s_i^l, s_i^m, s_i^u)$ değerleri ağırlıklandırılmış karar matrisindeki $\tilde{V}$ değerlerin toplamını temsil eder.

Adım 6: Fayda derecelerinin hesaplanması

11 ve 12 numaralı denklemler ile alternatiflerin fayda derecesi bulunur.

$$\widetilde{K_i^-} = \frac{\tilde{S}_i}{\tilde{S}_{ai}} = \left(\frac{s_i^l}{s_{ai}^u}, \frac{s_i^m}{s_{ai}^m}, \frac{s_i^u}{s_{ai}^l}\right) \quad (11)$$

$$\widetilde{K_i^+} = \frac{\tilde{S}_i}{\tilde{S}_{id}} = \left(\frac{s_i^l}{s_{id}^u}, \frac{s_i^m}{s_{id}^m}, \frac{s_i^u}{s_{id}^l}\right) \quad (12)$$

Adım 7: Yeni bulanık karar matrisinin oluşturulması ve durulaştırma işlemi.

Denklem 13 ile yeni bulanık matris $\tilde{T}_i$ oluştrulur.

$$\tilde{T}_i = \tilde{t}_i = (t_i^l, t_i^m, t_i^u) = \widetilde{K_i^-} \oplus \widetilde{K_i^+} = \left(k_i^{-l} + k_i^{+l}, k_i^{-m} + k_i^{+m}, k_i^{-u} + k_i^{+u}\right) \quad (13)$$

Daha sonra yeni bulanık sayı $\tilde{D}$ 14 numaralı denklem ile belirlenir.

$$\tilde{D} = (d^l, d^m, d^u) = \max_i \tilde{t}_{ij} \quad (14)$$

Son olarak bulanık sayı eşitlik 15 ile durulaştırılır.

$$\mathrm{df}_{\mathrm{crisp}} \frac{1+4\mathrm{m}+\mathrm{u}}{6} \quad (15)$$

Adım 8: İdeal ve en ideal olmayan çözüm fonksiyonlarının oluşturulması.

Denklem 16 ve 17 ile ideal $f(\widetilde{K_i^+})$ ve en ideal olmayan $f(\widetilde{K_i^-})$ çözümler bulunur.

$$f\left(\widetilde{K_i^+}\right) = \frac{\widetilde{K_i^-}}{\mathrm{df}_{\mathrm{crisp}}} = \left(\frac{k_i^{-l}}{df_{crisp}}, \frac{k_i^{-m}}{df_{crisp}}, \frac{k_i^{-u}}{df_{crisp}}\right) \quad (16)$$

$$f\left(\widetilde{K_i^-}\right) = \frac{\widetilde{K_i^+}}{\mathrm{df}_{\mathrm{crisp}}} = \left(\frac{k_i^{+l}}{df_{crisp}}, \frac{k_i^{+m}}{df_{crisp}}, \frac{k_i^{+u}}{df_{crisp}}\right) \quad (17)$$

Son olarak Adım 9 ile $f(\widetilde{K_i^+}), f(\widetilde{K_i^-}), \widetilde{K_i^-}, \widetilde{K_i^+}$ değerleri durulaştırılır.

Adım 9: Alternatiflerin fayda değerleri hesaplanır (Denklem 18).

$$f(\boldsymbol{K_i}) = \frac{K_i^+ + K_i^-}{1 + \frac{1-f(K_i^+)}{f(K_i^+)} + \frac{1-f(K_i^-)}{f(K_i^-)}} \quad (18)$$

Adım 10: Alternatifler fayda fonksiyonuna göre en iyiden en iyi olmayana doğru sıralanır.

Pay değeri yüksekliği en ideal çözüme işaret eder.

Literatüre bakıldığında MARCOS ve Bulanık MARCOS yönteminin birçok araştırma içerisinde kullanıldığını ve doğru sonuçlar verdiğini görmekteyiz. Tablo 4 içerisinde iki farklı yönteminde kullanıldığı çalışmalar özetlenmiştir:

Tablo 18: MARCOS ve Bulanık MARCOS ile yapılmış çalışmalar

Yazar(lar)	Yıl	Yöntem	Konu
Stević vd.	2020	MARCOS	Sağlık sektöründe sürdürülebilir tedarikçi seçimi
Puška vd.	2020	MARCOS	Proje yönetimi yazılımlarının değerlendirilmesi
Stanković vd.	2020	Bulanık MARCOS	Karayolu trafik riski analizi
Ilieva vd.	2020	Bulanık MARCOS	Bulut hizmeti sağlayıcı seçimi
Chakraborty vd.	2020	D-MARCOS	Demir-çelik sektöründe tedarikçi seçimi
Badi ve Pamucar	2020	Gri MARCOS	Çelik üretiminde tedarikçi seçimi
Ecer ve Pamucar	2021	Sezgisel Bulanık MARCOS	COVID-19 döneminde sigorta firmalarının performans analizi
Wang vd.	2022	Bulanık MARCOS (OPA entegrasyonlu)	Sürdürülebilir tedarikçi seçimi (Endüstri 4.0)
Ulutaş vd.	2020	Entegre MARCOS (CCSD-ITARA-MARCOS)	Lojistikte istifleme aracı seçimi
Martin vd.	2023	Nötrosofik MARCOS	Akıllı üretim sistemi için yazılım seçimi
El-Araby vd.	2024	MARCOS	Yer seçimi probleminde performans sıralaması

Kaynak: Yazar tarafından derlenmiştir.

Tablodaki çalışmalar, Bulanık MARCOS ve MARCOS yönteminin farklı alanlardaki uygulamalarını göstermektedir. Stević ve arkadaşları (2020), MARCOS yöntemini kullanarak sağlık sektöründe sürdürülebilir tedarikçi seçimi problemine çok kriterli bir çözüm sunmuşlardır. Puška ve ekibi (2020), çeşitli proje yönetimi yazılımlarını MARCOS yöntemiyle değerlendirerek en uygun yazılım alternatifini

belirlemişlerdir. Stanković ve çalışma arkadaşları (2020), karayolu trafik riski analizinde Bulanık MARCOS yöntemini uygulayarak belirsizlik ortamında güvenli rota seçimlerini incelemişlerdir. Ilieva ve ekibi (2020), bulut hizmet sağlayıcılarını bulanık değerlendirme matrisleriyle analiz ederek en uygun sağlayıcının seçimini gerçekleştirmiştir. Chakraborty ve arkadaşları (2020), demir-çelik sektöründe tedarikçi seçiminde D-sayılı MARCOS yöntemini kullanarak daha hassas bir alternatif sıralaması elde etmişlerdir. Badi ve Pamucar (2020), gri MARCOS yaklaşımıyla çelik üretiminde uygun tedarikçilerin seçilmesini sağlamışlardır. Ecer ve Pamucar (2021), sezgisel bulanık MARCOS yöntemiyle COVID-19 döneminde sağlık sigortası firmalarının performanslarını değerlendirmiştir. Wang ve meslektaşları (2022), Endüstri 4.0 çerçevesinde sürdürülebilir tedarikçi seçimi için OPA ve bulanık MARCOS yöntemlerini entegre ederek kapsamlı bir analiz sunmuşlardır. Ulutaş ve ekibi (2020), lojistik sektöründe istifleme ekipmanı seçiminde MARCOS'u diğer yöntemlerle entegre ederek karar sürecinin etkinliğini artırmışlardır. Martin ve çalışma arkadaşları (2023), akıllı üretim sistemlerinde kullanılacak yazılımların değerlendirilmesinde nötrosofik MARCOS yöntemini uygulamışlardır. Son olarak El-Araby ve ekibi (2024), yer seçimi probleminde MARCOS yöntemi ile alternatifleri karşılaştırarak performans sıralamasının sağlamlığını test etmişlerdir.

DÖRDÜNCÜ BÖLÜM
UYGULAMA

Bölüm içerisinde on farklı uzmandan alınan görüşler hibrit bir yaklaşım olan Geliştirilmiş Bulanık SWARA ve Bulanık MARCOS yöntemleri kullanılarak analiz edilecektir. Analiz sürecinde, Geliştirilmiş Bulanık SWARA yöntemi kriterlerin ağırlık değerlerini hesaplayarak hangi kriterin araç seçimi açısından daha öncelikli olduğunu belirleyecektir. Elde edilen bu ağırlık değerleri, Bulanık MARCOS yönteminde kullanılarak araçların birbirleriyle karşılaştırılması ve önem derecelerine göre sıralanması sağlanacaktır.

Karşılaştırılmak istenen araçlar yedi farklı kriter aracılığıyla değerlendirecektir. Seçilen kriterler:

- Maliyet: Aracın lisanslama modeli, kullanım ücreti ve uzun vadeli kullanımda ortaya çıkabilecek ekonomik yükü değerlendirilir. (Aracı kullanacak kişi bulma, yazılım güncelleme, başka araçlarla entegre etme vb.).
- Güvenlik: Aracın test süreçlerini güvenli şekilde yürütme kabiliyeti ve güvenlik açıklarına karşı direnci ele alınır.
- Raporlama Kalitesi: Test sonuçlarının anlaşılırlığı, ayrıntılı raporlama ve analiz yetenekleri incelenir.
- Dökümantasyon ve Topluluk Desteği: Araçların dökümantasyon yeterliliği, güncelliği ve kullanıcı topluluğunun araca olan etkin desteği ele alınır.
- Test Koşum Hızı: Otomasyon aracının test senaryolarını çalıştırma performansı ve hızı analiz edilir.
- Kararlılık (Stability): Aracın tutarlı sonuçlar üretme yeteneği ve farklı test ortamlarında istikrarlı çalışabilmesi incelenir.
- Araç Özellik Çeşitliliği: Paralel test koşumu, framework desteği, otomatik bekleme (wait) gibi gelişmiş özelliklerin varlığı ve çeşitliliği değerlendirilir.

Belirlenen kriterler, literatürde otomasyon aracı seçimine yönelik gerçekleştirilen önceki çalışmalara dayandırılarak oluşturulmuştur. Erol 2019'da yaptığı tez çalışmasında ele aldığı kriterler arasından maliyet ve dökümantasyon ile topluluk desteği kriterleri seçilmiştir. Kaymak 2020'de yaptığı tez içerisinde araçları karşılaştırmak için kullandığı kriterler arasından Test raporlama kalitesi, Test koşum

hızı ve araç özellik çeşitliliği kriterleri kullanılmak üzere belirlenmiştir. Son olarak Raulamo-Jurvanen ve 2016'da yaptığı çalışma arkadaşlarının içerisinden ise Kararlılık (Stability) ve Güvenlik kriterleri çalışma içerisinde kullanılmıştır. İncelenen geçmiş çalışmalarda yer alan tüm kriterler değerlendirmeye alınmamıştır; çünkü ilgili çalışmalarda tez içerisinde belirlenen kısıtlara uygun olmayan test otomasyon araçlarına yönelik bazı kriterler bulunmakta olup, söz konusu kriterler bu tezde karşılaştırılan açık kaynaklı ve kod tabanlı araçlar açısından önem taşımamaktadır.

Araç seçiminde kriterleri değerlendirmek için odak görüşmesi yapılan uzmanların bilgileri Tablo 19' da gösterilmiştir. **Araştırma kapsamında gerekli etik izinler, Dokuz Eylül Üniversitesi Sosyal ve Beşerî Bilimler Bilimsel Araştırma ve Yayın Etik Kurulu tarafından 22 Nisan 2025 tarihli ve 19 sayılı karar ile alınmıştır.**

Tablo 19: Uzman Bilgileri

Katılımcılar	Bölüm	Güncel Unvan	Sektör Tecrübesi
Uzman 1	Bilgisayar Mühendisliği	Test Otomasyon Takım lideri	16 yıl
Uzman 2	Bilişim sistemleri mühendisliği	Yazılım test lideri	10 yıl
Uzman 3	Bilgisayar Mühendisliği	Kalite Güvence direktörü	17 yıl
Uzman 4	Bilgisayar Mühendisliği	Yazılım test mühendisi takım lideri	10 yıl
Uzman 5	Elektronik mühendisliği	Kalite güvence mühendisi takım lideri	8 yıl
Uzman 6	Bilgisayar Mühendisliği	Uzman yazılım test mühendisi	13 yıl
Uzman 7	Bilgisayar Mühendisliği	Yazılım geliştirme yöneticisi	19 yıl
Uzman 8	Elektrik-elektronik mühendisliği	Uzman yazılım test mühendisi	8 yıl
Uzman 9	Bilgisayar programlama	Yazılım test takım lideri	14 yıl
Uzman 10	Bilgisayar Mühendisliği	Yazılım test yöneticisi	10 yıl

Kaynak: Yazar tarafından derlenmiştir.

Uzmanlar, sektörde minimum 8 maksimum 17 senelik tecrübe aralığında gerçek iş deneyimi olan ve test otomasyon alanında yönetici/direktör seviyesinde olan kişilerden seçilmiştir.

4.1 GELİŞTİRİLMİŞ BULANIK SWARA ÇÖZÜMÜ

Geliştirilmiş bulanık SWARA yönteminin ilk aşamasında uzmanlara kriterleri kendi içlerinde önem sırasına koymaları istenmiştir. En önemli kriterden en önemsize doğru azalan sıralama yapılması ve devamında daha önemli olan kriterle o kriterden sonra gelen kriterin arasında var olan önem derecelerini belirlemeleri talep edilmiştir. Kriterler arasındaki önem ilişkisini Tablo 20'deki dilsel değişkenlere göre yapılması belirtilmiştir.

Tablo 20. MARCOS Bulanık Sayılar

Dilsel Değişkenler	Bulanık Sayılar
Kesinlikle daha az önemli (ALS)	(1, 1, 1)
Ağırlıklı olarak daha az önemli (DLS)	(0.50, 0.67, 1)
Çok daha az önemli (MLS)	(0.40, 0.50, 0.67)
Gerçekten daha az önemli (RLS)	(0.33, 0.40, 0.50)
Daha az önemli (LS)	(0.29, 0.33, 0.40)
Orta derecede daha az önemli (MDLS)	(0.25, 0.29, 0.33)
Zayıf derecede daha az önemli (WLS)	(0.22, 0.25, 0.29)
Eşit derecede önemli (ES)	(0, 0, 0)

Kaynak: Vrtagic et al., 2021 alınan kaynaktan yazar tarafından Türkçeleştirilmiştir.

Kriterlerin sözel değerlere atanması ve tablo 20'de yer alan dilsel değişkenlere göre sıralanması işlemi Tablo 22'de temsili olarak gözlemlenebilir: (Tablo 22 gerçek uzman görüşü değildir, örnek olması açısından değerler eklenmiştir.)

Tablo 21. Kriterlerin İsmi

Kriter numarası	Kriter adı
C1	Maliyet
C2	Güvenlik
C3	Raporlama Kalitesi
C4	Doküman ve Topluluk Desteği
C5	Test Koşum Hızı

C6	Kararlılık
C7	Araç Özellik Çeşitliliği

Kaynak: Yazar tarafından derlenmiştir.

Tablo 21, araçların kendi aralarında sıralanması için uzmanlardan değerlendirilmesi istenen kriterleri gösterir.

Tablo 22. Örnek Kriter Değerlendirmesi

Önem Derecesi	Kriter	Değerlendirme
1	C2	-
2	C4	WLS
3	C6	RLS
4	C1	ALS
5	C3	MDLS
6	C5	MDLS
7	C7	ES

Kaynak: Yazar tarafından derlenmiştir.

4.1.1. Uzmanların Değerlendirmeleri

Odak görüşmesi yapılan uzmanların hepsi karşılaştırılmak istenen otomasyon araçlarıyla ilgili kriter değerlendirmelerini Tablo 22'de olduğu gibi gerçekleştirmiştir. Uzman 1 ve uzman 2'nin yaptığı değerlendirmeler sırasıyla Tablo 23 ve tablo 24'de gözlemlenebilir.

Tablo 23. Birinci Uzmana Ait Kriter Değerlendirme Tablosu

Önem Derecesi	Kriter	Değerlendirme
1	C6	-
2	C5	LS
3	C2	ES
4	C1	WLS
5	C7	MDLS

6	C3	MDLS
7	C4	ES

Kaynak: Yazar tarafından derlenmiştir

Birinci uzmana göre C6 kriteri en önemli kriterken C4 en az önemli kriterdir. C5 kriteri C6 kriterine göre daha az önemliyken (LS) C2 kriteri ise C5 ile eşit derecede önemlidir (ES). Diğer bütün kriterler kendinden bir önceki kriter ile karşılaştırılarak birbirleriyle olan önem dereceleri belirlenir.

Tablo 24: İkinci Uzmana Ait Kriter Değerlendirme Tablosu

Önem Derecesi	Kriter	Değerlendirme
1	C6	-
2	C4	WLS
3	C1	ES
4	C7	LS
5	C3	LS
6	C5	LS
7	C2	LS

Kaynak: Yazar tarafından derlenmiştir

4.1.2. Kriter Ağırlıklandırma Tablosu

Uzmanlardan elde edilen değerlendirme sonuçları ilk olarak Tablo 2'de yer alan bulanık değerlerle ifade edilmekte ve devamında her bir uzmana ait kriter katsayılarının hesaplanması, ağırlıkların bulanık olarak yeniden belirlenmesi, durulaştırılması ve normalize edilmesi adımları bütün uzmanların görüşleri için hesaplanmaktadır. Son olarak tüm uzmanların bireysel görüşleri ile elde edilen bulanık ve normalize edilmiş ağırlıklar birleştirilmiş ve bu değerlerin ortalaması alınarak her kriter için nihai ağırlıklar hesaplanmıştır. Böylelikle uzman görüşlerinin ortak değerlendirilmesiyle kriterlerin genel ağırlıkları ve önem sıralaması belirlenmiştir. Tablo 25 ve Tablo 26 Birinci ve ikinci uzman için normalize ağırlık tablosudur, hesaplanan bulanık ağırlık değerleri MARCOS yönteminde kullanılacaktır. Tablo 27

ve 28'de ise uzman 1 ve 2'nin durulaştırılan bulanık ağrılık değerleri normalize edilerek normalize edilmiş ağırlık değerleri hesaplanır. Son olarak Tablo 11 üzerinde bütün uzmanların normalize edilmiş ağırlık değerlerinin kapsamlı olarak değerlendirilmesi yapılır ve sonucunda kriterlerin önem sırasına göre sıralanır.

Tablo 25: Birinci Uzmana Ait Bulanık Ağırlık Değerler Tablosu

	Bulanık ifad			Ağırlık katsayısı			Yeniden hesaplanan ağırlık değerleri			Bulanık ağırlık değerleri		
C6				1	1	1	1	1	1	0,222	0,233	0,248
C5	0,29	0,33	0,4	1,29	1,33	1,40	0,714	0,752	0,775	0,158	0,175	0,193
C2	0	0	0	1,00	1,00	1,00	0,714	0,752	0,775	0,158	0,175	0,193
C1	0,22	0,25	0,29	1,22	1,25	1,29	0,554	0,602	0,635	0,123	0,140	0,158
C7	0,25	0,29	0,33	1,25	1,29	1,33	0,416	0,466	0,508	0,092	0,109	0,126
C3	0,25	0,29	0,33	1,25	1,29	1,33	0,313	0,361	0,407	0,069	0,084	0,101
C4	0,00	0	0,00	1,00	1,00	1,00	0,313	0,361	0,407	0,069	0,084	0,101

Kaynak: Yazar tarafından derlenmiştir.

Tablo 26: İkinci Uzmana Ait Bulanık Ağırlık Değerler Tablosu

	Bulanık ifade			Ağırlık katsayısı			Yeniden hesaplanan ağırlık değerleri			Bulanık ağırlık değerleri		
C6				1	1	1	1	1	1	0,225	0,235	0,251
C4	0,22	0,25	0,29	1,22	1,25	1,29	0,775	0,800	0,820	0,174	0,188	0,206
C1	0	0	0	1,00	1,00	1,00	0,775	0,800	0,820	0,174	0,188	0,206
C7	0,29	0,33	0,40	1,29	1,33	1,40	0,554	0,602	0,635	0,125	0,142	0,159
C3	0,29	0,33	0,40	1,29	1,33	1,40	0,396	0,452	0,493	0,089	0,106	0,124
C5	0,29	0,33	0,40	1,29	1,33	1,40	0,283	0,340	0,382	0,064	0,080	0,096
C2	0,29	0,33	0,40	1,29	1,33	1,40	0,202	0,256	0,296	0,045	0,060	0,074

Kaynak: Yazar tarafından derlenmiştir

Tablo 27: Birinci Uzmana Ait Durulaştırılan Ağırlık Değerler Tablosu

Durulaştırılan değerler	Normalize ağırlıklar
0,234	**0,233**
0,175	**0,175**
0,175	**0,175**
0,140	**0,140**
0,109	**0,109**
0,085	**0,084**
0,085	**0,084**
	1,000

Kaynak: Yazar tarafından derlenmiştir.

Tablo 28: İkinci Uzmana Ait Durulaştırılan Ağırlık Değerler Tablosu

Durulaştırılan değerler	Normalize ağırlıklar
0,236	**0,236**
0,189	**0,188**
0,189	**0,188**
0,142	**0,141**
0,106	**0,106**
0,080	**0,080**
0,060	**0,060**
	1,000

Kaynak: Yazar tarafından derlenmiştir.

Tablo 29: Normalize Edilen Nihai Ağırlık Tablosu

	Uzman 1	**Uzman 2**	**...**	**Uzman 10**	**Geometrik Ortalama**	**Final Ağırlıkları**	**Sıralama**	
C1	0,140	0,188	...	0,121	0,130	0,139	**5**	Maliyet
C2	0,175	0,060	...	0,243	0,136	0,145	**3**	Güvenlik
C3	0,084	0,106	...	0,071	0,107	0,115	**7**	Raporlama kalitesi
C4	0,084	0,188	...	0,094	0,144	0,154	**2**	Döküman ve topluluk desteği
C5	0,175	0,080	...	0,156	0,133	0,142	**4**	Test koşum hızı
C6	0,233	0,236	...	0,195	0,163	0,174	**1**	Stability
C7	0,109	0,141	...	0,121	0,122	0,131	**6**	Araç özellik çeşitliliği

Kaynak: Yazar tarafından derlenmiştir.

Tablo 29'da görüldüğü üzere otomasyon aracının kararlı çalışması 0.174'lük katsayı ile en önemli kriterdir. En az önemli kriter ise uzmanların ortak görüşüyle aracın raporlama kalitesi olarak belirlenmiştir. Genel anlamda sıralamaya bakıldığında otomasyon aracının neler yapabildiğinden ziyade düzgün ve kaliteli çalışabilmesi uzmanlar için daha önemli olarak yorumlanabilir çünkü kararlılık kriterinden hemen sonra iki numaralı önemli kriteri doküman ve topluluk desteği olarak ve ardından güvenlik kriteri olarak seçildiğini görmekteyiz.

4.2 BULANIK MARCOS ÇÖZÜMÜ

Bulanık MARCOS yönteminin uygulanmasında ilk adım olarak, karar vericilerden Tablo 3'te verilen sözel değişkenlere göre araçları kriterler açısından ayrı ayrı değerlendirmeleri istenmektedir. Araçlara ilişkin yapılan değerlendirmeler, yine Tablo 3'te bulunan bulanık sayı değerlerine dönüştürülerek başlangıç karar matrisi oluşturulur. İkinci adımda, ideal ve ideal olmayan çözümler temel alınarak bulanık genişletilmiş karar matrisi elde edilir. Devamında normalizasyon işlemi gerçekleştirilerek normalize karar matrisi oluşturulur. Sonraki aşamada, Geliştirilmiş Bulanık SWARA yöntemi ile belirlenen kriterlerin bulanık ağırlık değerleri kullanılarak ağırlıklandırılmış bulanık karar matrisi meydana getirilir. Elde edilen bu matristeki değerler temel alınarak bulanık karar matrisi hesaplanır ve ardından alternatiflerin fayda dereceleri belirlenir. Yöntemin son aşamasında bulanık T matrisi oluşturulur ve bu matristeki en ideal çözüm ile ideal olmayan çözüm fonksiyonları hesaplanır. Bulunan çözüm fonksiyonları yardımıyla alternatiflerin fayda değerleri elde edilerek, pay değerlerine göre alternatiflerin ideal olandan ideal olmayana doğru sıralaması yapılır.

Tablo 30 ve tablo 31 üzerinde sırasıyla uzman 1 ve uzman 2'nin yaptığı değerlendirmeler gözlemlenir. Yapılan bütün uzman değerlendirmeleriyle birlikte başlangıç bulanık karar matrisi oluşturulur. (Tablo 32):

Tablo 30: Uzman 1 Alternatif Değerlendirmeleri

Kriterler / Araçlar	C1	C2	C3	C4	C5	C6	C7
Selenium	Çİ	İ	İ	O	İ	Çİ	O
Cypress	O	İ	Çi	İ	İ	İ	Çİ
Playwright	OK	O	İ	İ	İ	İ	Çİ
Puppeterr	İ	O	İ	O	İ	İ	O

Kaynak: Yazar tarafından derlenmiştir.

Tablo 31: Uzman 2 Alternatif Değerlendirmeleri

Kriterler / Araçlar	C1	C2	C3	C4	C5	C6	C7
Selenium	Çİ	İ	İ	İ	İ	Çİ	İ
Cypress	Çİ	İ	O	Çİ	Çİ	O	Çİ
Playwright	Çİ	İ	İ	İ	İ	Çİ	İ
Puppeterr	Çİ	İ	O	İ	Çİ	O	İ

Kaynak: Yazar tarafından derlenmiştir.

Tablo 32: Başlangıç Bulanık Karar Matrisi

	A1			A2			A3			A4		
C1	4,00	6,40	7,00	3,00	4,70	7,00	1,00	5,20	7,00	2,00	5,50	7,00
C2	2,00	5,00	7,00	3,00	5,60	7,00	1,00	5,20	7,00	2,00	4,40	7,00
C3	1,00	5,10	7,00	1,00	5,70	7,00	3,00	5,90	7,00	2,00	4,60	7,00
C4	3,00	6,50	7,00	3,00	5,80	7,00	1,00	5,30	7,00	1,00	4,60	7,00
C5	2,00	5,20	7,00	5,00	6,30	7,00	5,00	6,40	7,00	2,00	5,80	7,00
C6	2,00	5,30	7,00	2,00	5,60	7,00	3,00	6,10	7,00	1,00	5,10	7,00
C7	3,00	6,30	7,00	3,00	5,90	7,00	3,00	6,10	7,00	3,00	4,40	7,00

Kaynak: Yazar tarafından derlenmiştir.

Tablo 32 üzerinde A değerleri alternatif test araçlarını temsil etmektedir (A1: Selenium, A2: Cypress, A3: Playwright, A4: Puppeteer). En ideal (A(AI)) ve en ideal olmayan (A(ID)) çözümler ele alınarak bulanık genişletilmiş karar matrisi oluşturulur ve devamında matris ile birlikte oluşturulan değerlerin yardımıyla normalize karar matrisi hesaplanır.

Tablo 33: Bulanık Genişletilmiş Karar Matrisi

	A1			A2			A3			A4			A(AI)			A(ID)		
C1	4,00	6,40	7,00	3,00	4,70	7,00	1,00	5,20	7,00	2,00	5,50	7,00	1,00	4,70	7,00	4,00	6,40	7,00
C2	2,00	5,00	7,00	3,00	5,60	7,00	1,00	5,20	7,00	2,00	4,40	7,00	1,00	4,10	7,00	3,00	5,60	7,00
C3	1,00	5,10	7,00	1,00	5,70	7,00	3,00	5,90	7,00	2,00	4,60	7,00	1,00	4,60	7,00	3,00	5,90	7,00
C4	3,00	6,50	7,00	3,00	5,80	7,00	1,00	5,30	7,00	1,00	4,60	7,00	1,00	4,60	7,00	3,00	6,50	7,00
C5	2,00	5,20	7,00	5,00	6,30	7,00	5,00	6,40	7,00	2,00	5,80	7,00	2,00	3,00	7,00	5,00	7,00	7,00
C6	2,00	5,30	7,00	2,00	5,60	7,00	3,00	6,10	7,00	1,00	5,10	7,00	1,00	5,10	7,00	3,00	6,10	7,00
C7	3,00	6,30	7,00	3,00	5,90	7,00	3,00	6,10	7,00	3,00	4,40	7,00	3,00	4,40	7,00	3,00	6,30	7,00

Kaynak: Yazar tarafından derlenmiştir.

Tablo 34: Normalize Karar Matrisi

	A1			A2			A3			A4			A(AI)			A(ID)		
C1	0,44	0,71	0,78	0,33	0,52	0,78	0,11	0,58	0,78	0,22	0,61	0,78	0,11	0,52	0,78	0,44	0,71	0,78
C2	0,22	0,56	0,78	0,33	0,62	0,78	0,11	0,58	0,78	0,22	0,49	0,78	0,11	0,49	0,78	0,33	0,62	0,78
C3	0,14	0,73	1,00	0,14	0,81	1,00	0,33	0,66	0,78	0,22	0,51	0,78	0,14	0,66	1,00	0,43	0,84	1,00
C4	0,33	0,72	0,78	0,33	0,64	0,78	0,11	0,59	0,78	0,11	0,51	0,78	0,11	0,51	0,78	0,33	0,72	0,78
C5	0,22	0,33	0,78	0,56	0,78	0,78	0,56	0,56	0,78	0,22	0,56	0,78	0,22	0,33	0,78	0,56	0,78	0,78
C6	0,22	0,59	0,78	0,22	0,62	0,78	0,33	0,68	0,78	0,11	0,57	0,78	0,11	0,57	0,78	0,33	0,68	0,78
C7	0,33	0,70	0,78	0,33	0,66	0,78	0,33	0,68	0,78	0,33	0,49	0,78	0,33	0,49	0,78	0,33	0,70	0,78

Kaynak: Yazar tarafından derlenmiştir.

Geliştirilmiş Bulanık SWARA yönteminde elde edilen bulanık kriter ağırlık değerleri (Tablo 5) çözüme dahil edilerek ağırlıklandırılmış bulanık karar matrisi oluşturulur (Tablo 36). Oluşturulan matris üzerindeki değerler toplanarak alternatiflerin fayda derecelerini belirlemede kullanılacak en son bulanık karar matrisi (S_i) Tablo 37'de olduğu gibi hesaplanır.

Tablo 35: Bulanık Ağırlık Değerleri

	WJ		
C1	0,114	0,130	0,146
C2	0,121	0,136	0,151
C3	0,092	0,107	0,123
C4	0,130	0,143	0,158
C5	0,116	0,133	0,149
C6	0,149	0,163	0,177
C7	0,106	0,122	0,138

Kaynak: Yazar tarafından derlenmiştir.

Tablo 36: Ağırlıklandırılmış Bulanık Karar Matrisi

	A1			A2			A3			A4			A(AI)			A(ID)		
C1	0,051	0,092	0,114	0,038	0,068	0,114	0,013	0,075	0,114	0,025	0,079	0,114	0,013	0,068	0,114	0,051	0,092	0,114
C2	0,027	0,075	0,094	0,040	0,084	0,117	0,013	0,078	0,117	0,027	0,066	0,117	0,013	0,066	0,117	0,040	0,084	0,117
C3	0,013	0,078	0,123	0,013	0,087	0,123	0,031	0,070	0,096	0,020	0,055	0,096	0,013	0,071	0,123	0,040	0,091	0,123
C4	0,043	0,104	0,123	0,043	0,092	0,123	0,014	0,085	0,123	0,014	0,073	0,123	0,014	0,073	0,123	0,043	0,104	0,123
C5	0,026	0,010	0,116	0,065	0,023	0,116	0,065	0,016	0,116	0,026	0,016	0,116	0,026	0,010	0,116	0,065	0,023	0,116
C6	0,033	0,096	0,138	0,033	0,101	0,138	0,050	0,110	0,138	0,017	0,092	0,138	0,017	0,092	0,138	0,050	0,110	0,138
C7	0,035	0,085	0,107	0,035	0,080	0,107	0,035	0,083	0,107	0,035	0,060	0,107	0,035	0,060	0,107	0,035	0,085	0,107

Kaynak: Yazar tarafından derlenmiştir.

Tablo 37: Nihai Bulanık Karar Matrisi

	S_i		
A(AI)	0,132	0,440	0,837
A1	0,228	0,540	0,814
A2	0,268	0,536	0,837
A3	0,221	0,517	0,810
A4	0,165	0,442	0,810
A(ID)	0,324	0,589	0,837

Kaynak: Yazar tarafından derlenmiştir.

Son aşamada alternatiflerin fayda dereceleri (K^{+}_{i} ve K^{-}_{i}) hesaplanmakta olup, fayda fonksiyonlarının belirlenmesi için gerekli olan T_i matrisi oluşturulur. Elde edilen fayda fonksiyonları ve fayda derecelerinden yararlanılarak, alternatiflerin nihai sıralaması, her bir alternatife ait fonksiyonun pay değerleri temel alınarak gerçekleştirilmektedir.

Tablo 38: Alternatiflerin İdeal Fayda Derecesi

	K^{+}_{i}			
A1	0,278	0,924	2,573	1,0909
A2	0,327	0,907	2,573	1,0879
A3	0,273	0,911	2,573	1,0816
A4	0,204	0,777	2,573	0,9808

Kaynak: Yazar tarafından derlenmiştir.

Tablo 39: Alternatiflerin Anti-ideal Fayda Derecesi

	K^{-}_{i}			
A1	0,278	1,235	6,296	1,919
A2	0,327	1,213	6,296	1,912
A3	0,273	1,218	6,296	1,907
A4	0,204	1,039	6,296	1,776

Kaynak: Yazar tarafından derlenmiştir.

Tablo 40: Ti Matrisi

A1	0,557	2,159	8,869
A2	0,655	2,120	8,869
A3	0,546	2,129	8,869
A4	0,407	1,816	8,869
MAX	0,655	2,120	8,869

Kaynak: Yazar tarafından derlenmiştir.

Tablo 41: İdeal Fayda Fonksiyonu f(k+)

A1	0,093	0,412	2,098	0,6397
A2	0,109	0,404	2,098	0,6374
A3	0,091	0,406	2,098	0,6356
A4	0,068	0,346	2,098	0,592

Kaynak: Yazar tarafından derlenmiştir

Tablo 42: Anti-ideal Fayda Fonksiyonu f(k-)

A1	0,093	0,308	0,857	0,3636
A2	0,109	0,302	0,857	0,3626
A3	0,091	0,304	0,857	0,3605
A4	0,068	0,259	0,857	0,3269

Kaynak: Yazar tarafından derlenmiştir

Tablo 43: Nihai Sıralama Tablosu

	Pay	Payda	$F(k_i)$
A1	3,010	3,314	0,908
A2	3,000	3,327	0,902
A3	2,989	3,347	0,893
A4	2,757	3,748	0,735

Kaynak: Yazar tarafından derlenmiştir

Nihai tablodaki pay değerleri incelendiğinde, alternatiflerin en iyiden başlayarak sıralaması sırasıyla; A1 > A2 > A3 > A4 şeklindedir. Buna göre en iyi alternatif Selenium (A1), ardından Cypress (A2), Playwright (A3) ve son olarak Puppeteer (A4) olarak belirlenmiştir.

SONUÇ

Tez çalışması içerisinde, web tabanlı yazılım testi otomasyonu geliştirme süreçlerinde kullanılan popüler dört farklı test otomasyon aracı belirlenen kriterlere dayandırılarak karşılaştırmalı olarak incelenmiş ve karma ÇKKV yöntemleri kullanılarak değerlendirilmiştir.

Çalışmanın temel amacı, web yazılımı sektöründeki yazılım geliştiriciler, test mühendisleri ve karar vericiler için en uygun test otomasyon aracının seçim sürecini kolaylaştırmak ve böylelikle otomasyon test süreçlerinde etkinlik ve kaliteyi artırmaktır. Çalışma amacı kapsamında, otomasyon süreçlerinde yapılacak iyileştirmelerin doğrudan yazılımın başarısını olumlu yönde etkilemesi ve maddi olarak yüksek miktarda zararın önüne geçmesi beklenir.

Araştırma kapsamında, otomasyon araçlarının değerlendirilmesinde kullanılan 7 farklı kriter literatüre dayandırılarak seçilmiş ve alanında uzman 10 test otomasyon mühendisinin görüşleri ile değerlendirilmiştir. Kriterlerin değerlendirilmesinde Geliştirilmiş Bulanık SWARA yöntemi kullanılarak her kriterin öncelik sırası ve ağırlık değerleri belirlenmiştir. Uzmanların görüşleri neticesinde, "Kararlılık (Stability)" kriterinin en yüksek önceliğe sahip olduğu, bunu sırasıyla "Dokümantasyon ve Topluluk Desteği" ve "Güvenlik" kriterlerinin takip ettiği görülmüştür. Araçların ne yaptığından çok, tutarlı ve güvenilir sonuçlar üretmelerinin uzmanlar için daha kritik olduğu anlaşılmıştır. Belirlenen kriter ağırlıkları Bulanık MARCOS yöntemiyle otomasyon araçlarının karşılaştırılmasında kullanılmıştır. Yapılan analiz sonucunda, değerlendirmeye alınan dört araçtan Selenium'un en yüksek performansı gösterdiği belirlenmiştir. Selenium'u sırasıyla Cypress, Playwright ve Puppeteer araçları izlemiştir. Elde edilen sonuçlar, sektör uygulayıcılarına araç seçiminde bilimsel olarak destek sağlayacak nitelikte olup, araçların etkin kullanımı ve olası maddi-manevi kayıpların engellenmesi açısından önemli katkılar sunmaktadır. Cypress ve Selenium araçlarının birbirlerine çok yakın (3,000 ve 3,010) değerlere sahip olması aralarında genel performansları bakımından yüksek derecede farklar olmadığını ifade etmektedir.

Araştırmanın örneklemi, web test otomasyonu alanında uzman/yönetici pozisyonlarında aktif görev alan 10 farklı bireyden oluşmaktadır. Ele alınan örneklem

sayısı daha önce yapılmış benzer çalışmalara dayandırılarak belirlenmiştir fakat diğer çalışmalardan farklı olarak seçilen kişilerin tecrübesi 8 ila 17 yıl arasında değişkenlik göstermektedir. Uzmanların yüksek tecrübe düzeyi araştırma sonuçlarını daha tutarlı ve güvenilir kılmaktadır.

Elde edilen sonuçlar, literatürde yer alan benzer çalışmalarla karşılaştırıldığında tutarlılık göstermektedir. Sharma ve Angmo 2014'de yaptığı çalışmaya göre Selenium web otomasyonu için en iyi otomasyon aracı olarak belirlenmiştir. Günümüzde ise değişen ve gelişen teknoloji ile birlikte yeni otomasyon araçları ortaya çıkmış ve özellikle Cypress aracının kullanıcı dostu yapısı, kararlılığı, güçlü topluluk desteği ve test koşum hızı ile literatürdeki diğer araçların öne çıkan özelliklerini bünyesinde barındırdığı ve hatta bazı noktalarda üstünlük sağladığı görülmüştür. Bunun sonucu olarak çalışma sonunda Cypress ve Selenium aracı arasında yüksek bir performans farkı ortaya konmamıştır.

Gelecekte yapılacak çalışmalar için, araç seçiminde yapay zeka sistemlerinin ve daha yüksek miktarda örneklem sayısının kullanılması önerilmektedir. Böylece daha fazla uzman görüşü denkleme dahil olacak ve yapay zekanın objektif bakış açısı ile uzmanların subjektif fikirlerinin birleşimi gerçekleşebilecektir.

Sonuç olarak, web tabanlı yazılımların başarısında yazılım test süreçlerinin etkin yönetiminin kritik bir rol oynadığı ve test süreçlerinin etkinliği için uygun test otomasyon araçlarının doğru biçimde seçilmesinin gerektiği ifade edilebilir. Yazılım sektörünün sürekli gelişmesi ve büyümesi, yazılım sistemlerinin karmaşıklığını artırmakta, bu durum da test süreçlerinin önemini ve kapsamını genişletmektedir. Yazılım test süreçlerinin, sektörün gelişim hızına paralel olarak sürekli güncellenmesi ve iyileştirilmesi zorunlu hale gelmiştir. Bu bağlamda, firmaların yazılım testi alanındaki yatırımlarını artırmaları gerektiği ve gelecekte bu alandaki yatırım ve çalışmaların daha da yoğunlaşacağı öngörülmektedir.

KAYNAKÇA

Abbas, R., Sultan, Z., ve Bhatti, S. N. (2017). Comparative analysis of automated load testing tools: Apache jmeter, microsoft visual studio (tfs), loadrunner, siege. International conference on communication Technologies. (ss. 39-44). IEEE.

Abdulla, A., Bargiannis, G. ve Badi, I. Effective Supplier Selection Combining Machine Learning with the Marcos Method. Decision Analytics Journal. 9: 100342.

Aburas, A. (2024). Choosing the Right Automated Software Testing Tools. In 2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA). (ss. 31-35). IEEE.

Akbaş, Ö.Y. (2016). Yazılım Test Otomasyonu. Dokuz Eylül Üniversitesi Araştırma ve İnovasyon Dergisi. 3(9): 36-39.

Alferidah, S. K., ve Ahmed, S. (2020). Automated software testing tools. In 2020 International Conference on Computing and Information Technology (ICCIT-1441) (ss. 1-4). IEEE.

Amazon, (2023). aws.amazon, Web3 nedir?. (01.02.2025).

Aytekin, A. ve Gündoğdu, H. G. (2021). OECD ve AB Üyesi Ülkelerin Sürdürülebilir Yönetişim Düzeylerine Göre Swara Tabanlı Topsıs-Sort-B Ve Waspas Yöntemleriyle İncelenmesi. Öneri Dergisi. 16(56): 943-971.

Aşan, H., ve Ayçin, E. (2020). Kurumsal kaynak planlama sistemlerinin seçimindeki kriterlerin best-worst metodu ile değerlendirilmesi. Akademik İzdüşüm Dergisi. 5(2): 114-124.

Badi, I. ve Pamucar, D. (2020). Supplier selection for steelmaking company by using combined Grey-MARCOS methods. Decision Making: Applications in Management and Engineering. 3(2): 37-48.

Belton, V. ve Stewart, T.J. (2002). The Multiple Criteria Problem. Multiple Criteria Decision Analysis (pp.13-33). Boston: Springer.

Browserstack, (13.03.2025). BrowserStack, What is Test Automation: Benefits, Limitations, Tools, and Best Practices. . (28.03.2025).

Castells, M. (2010). Globalisation, networking, urbanisation: Reflections on the spatial dynamics of the information age. Urban studies. 47(13), 2737-2745.

Cen, Z. ve Zhao, Y. (2024). Enhancing User Engagement through Adaptive Interfaces: A Study on Real-time Personalization in Web Applications. Journal of Economic Theory and Business Management. 1: 1-7.

Clarke, J. (2012). What Is SQL Injection?. SQL Injection Attacks and Defense (pp.2-27). Massachusetts: Syngress Publishing.

Coderspace, (14.10.2024). coderspace, Yazılım Test Yaşam Döngüsü Nedir? | STLC Aşamaları.. (02.02.2025).

Coderspace. (14.10.2024). Yazılım Testi Nedir?. https://coderspace.io/blog/yazilim-testi-nedir/#header-index-16, (03.02.2025).

Crowther, S. (04.02.2021). What is Puppeteer? An Automation Framework Built by Google. . (25.03.2025).

Cypress.io (2020), Testing is the Key to Continuous Innovation. . (22.03.2025).

Cypress.io (2020), Why Cypress?. https://docs.cypress.io/app/get-started/why-cypress. (22.03.2025).

Deo, T.K. (07.03.2023). Lambdatest, Selenium WebDriver Architecture Explained – Ultimate Guide On What Is Selenium Architecture And How Does It Work. https://www.lambdatest.com/blog/selenium-webdriver-architecture/. (10.02.2025).

Duc Trung, D. (2022). Multi-criteria decision making under the MARCOS method and the weighting methods: applied to milling, grinding and turning processes. Manufacturing Review. 9: 3.

Ecer, F. ve Pamucar, D. (2021). MARCOS technique under intuitionistic fuzzy environment for determining the COVID-19 pandemic performance of insurance companies in terms of healthcare services. Applied Soft Computing. 104: 107199.

Erdem, A., Kara, M. A. ve Dumlu, H. (2023). Dijital Platform Tercihlerinin Swara Ve Gri İlişkisel Analiz Yöntemleri İle Değerlendirilmesi. Uluslararası İktisadi ve İdari İncelemeler Dergisi. (41): 91-106.

Erol, E. (2019). Yazılım test otomasyon araçlarının karşılaştırılması. İstanbul: İstanbul Üniversitesi-Cerrahpaşa Lisansüstü Eğitim Enstitüsü.

Ersöz, B. (2020). Yeni Nesil Web Paradigması-Web 4.0. Bilgisayar Bilimleri ve Teknolojileri Dergisi. 1(2): 58-65.

Ersöz, B. (2020). Yeni nesil web paradigması-web 4.0. Bilgisayar Bilimleri ve Teknolojileri Dergisi. 1(2): 58-65.

Eşitti, Ş. ve Işık, M. (2015). Sosyal Medyanın Yabancı Turistlerin Türkiye'yi Tatil Destinasyonu Olarak Tercih Etmelerine Etkisi. Karadeniz Uluslararası bilimsel dergisi. 1(27): 11-33.

Gamido, H. V., Gamido, M. V., ve Sison, A. M. (2019). Developing a secured image file management system using modified AES. Bulletin of Electrical Engineering and Informatics. 8(4): 1461-1467.

Garousi, V., Coşkunçay, A., Betin-Can, A. ve Demirörs, O. (2013). Türkiye'deki yazılım test uygulamaları anketi. Ulusal Yazılım Mühendisliği Sempozyumu (UYMS 2013), CEUR Workshop Proceedings, 1072, 1–10.

Garousi, V., ve Mäntylä, M. V. (2016). When and what to automate in software testing? A multi-vocal literature review. Information and Software Technology. 76, 92-117.

Harmancı, F. M., Emektar, M., Öncü, S. ve Güngör, N. G. (2025). Test otomasyonunda yaşanan büyük zorluklar ve uzmanların tavsiyeleri: sektörden uzman değerlendirmeleri. Kamu Yönetimi ve Teknoloji Dergisi. 7(1), 103-118.

He, J., Huang, Z., Mishra, A. R. ve Alrasheedi, M. (2021). Developing a new framework for conceptualizing the emerging sustainable community-based tourism using an extended interval-valued Pythagorean fuzzy SWARA-MULTIMOORA. Technological Forecasting and Social Change. 171: 120955.

Hypernet, (16.06.2022), Web 3.0 Nedir?. . (01.02.2025).

Ilieva, G., Yankova, T., Hadjieva, V., Doneva, R. ve Totkov, G. (2020). Cloud service selection as a fuzzy multi-criteria problem. TEM Journal. 9(2): 484.

Jeremić, V., Bucea-Manea-Ţonis, R., Vesić, S., ve Stefanović, H. (2023). Revolutionizing Software Testing: The Impact of AI, ML, and IoT. EdTech Journal. 3(1): 12-15.

Kapan, K. ve Üncel, R. (2020). Gelişen Web Teknolojilerinin (Web 1.0- Web 2.0-Web 3.0) Türkiye Turizmine Etkisi. Safran Kültür ve Turizm Araştırmaları Dergisi. 3(3): 276-289.

Kaymak, A. (2020). Comparison of software testing tools and selection of automation over manual testing. Ankara: Yıldırım Beyazıt Üniversitesi-Fen Bilimleri Enstitüsü.

Kertusha, I., Assress, G., Duman, O. Ve Arcuri, A. (2025). A Survey on Web Testing: On the Rise of AI and Applications in Industry. arXiv preprint arXiv:2503.05378

Kertusha, I., Assress, G., Duman, O. ve Arcuri, A. (2025). A Survey on Web Testing: On the Rise of AI and Applications in Industry. arXiv preprint arXiv:2503.05378.

Keshavarz Ghorabaee, M., Zavadskas, E. K., Olfat, L., ve Turskis, Z. (2018). A new extension of the fuzzy SWARA method for evaluating criteria using trapezoidal fuzzy numbers. Technological and Economic Development of Economy, 24(3), 942–958.

Keshavarz-Ghorabaee, M., Amiri, M., Zavadskas, E. K., Turskis, Z., ve Antucheviciene, J. (2021). Determination of Objective Weights Using a New Method Based on the Removal Effects of Criteria (MEREC). Symmetry, 13(4): 525.

Keshavarz-Ghorabaee, M., Amiri, M., Zavadskas, E. ve Antucheviciene, J. (2018). An extended SWARA method for determining the objective weights of criteria in multi-criteria decision-making problems. Symmetry. 10(4): 91.

KHALEDALSAWAF, M. (2019). Evaluation and comparison of web-based automated testing tools. İstanbul: İstanbul Üniversitesi-Cerrahpaşa Lisansüstü Eğitim Enstitüsü.

Kılıç, B. N. ve Asal, Ö. (2024). Optimal Trafik Denetim Stratejilerinin Belirlenmesinde Çok Kriterli Karar Verme Yaklaşımları. Kent Akademisi (Online). 17(2): 551-567.

Kırık, A. M. (2013). Gelişen web teknolojileri ve sosyal medya bağımlılığı. Sosyal medya araştırmaları (pp. 77-101). Konya: Çizgi Kitabevi.

Li, T., Huang, R., Cui, C., Towey, D., Ma, L., Li, Y.-F., ve Xia, W. (2024). A survey on web application testing: A decade of evolution. arXiv preprint arXiv:2412.10476.

Mardani, A., Zavadskas, E. K., Govindan, K. ve Nilashi, M. (2017). Multiple criteria decision-making techniques in transportation systems: a systematic review of the state of the art literature. Transport Policy. 59: 17-29.

Martin, N., Broumi, S., Sudha, S. ve Priya, R. (2023). Neutrosophic marcos in decision making on smart manufacturing system. Neutrosophic Systems with Applications. 4(1): 12-32.

Melkozerova, O., Nariezhnii, O., ve Malakhov, S. (2019). Analysis of tools for automated software testing. Computer Science and Cybersecurity. 1: 75-84.

Mobaraya, F., ve Ali, S. (2019, December). Technical Analysis of Selenium and Cypress as functional automation framework for modern web application testing. In 9th international conference on computer science. (ss. 27-46), Department of Information Technology, AGI Institute, Auckland, New Zealand.

Npm, (2022). nmptrends, cypress vs playwright vs puppeteer vs selenium-webdriver. . (09.02.2025).

Pelivani, E., Besimi, A., ve Cico, B. (2022). A comparative study of UI testing framework. In 2022 11th Mediterranean Conference on Embedded Computing (MECO). (ss. 1-5). IEEE.

Pelivani, E., ve Cico, B. (2021, June). A comparative study of automation testing tools for web applications. In 2021 10th Mediterranean Conference on Embedded Computing (MECO). (ss. 1-6). IEEE.

Playwright.dev (2023). Playwright Library. . (28.03.2025).

Pptr.dev (2023), What is Puppeteer?. . (25.03.2025).

Puška, A., Stević, Ž. ve Pamučar, D. (2022). Evaluation and selection of healthcare waste incinerators using extended sustainability criteria and multi-criteria analysis methods. Environment, Development and Sustainability. 1-31.

Puška, A., Stojanović, I., Maksimović, A. ve Osmanović, N. (2020). Project management software evaluation by using the measurement of alternatives and ranking according to compromise solution (MARCOS) method. Operational Research in Engineering Sciences: Theory and Applications. 3(1): 89-102.

Sabev, P., ve Grigorova, K. (2017). A comparative study of GUI automated tools for software testing. In Proc. of The Third International Conference on Advances and Trends in Software Engineering. (ss. 7-15). SOFTENG 2017.

Saeidi, P., Mardani, A., Mishra, A. R., Cajas Cajas, V. E. ve Carvajal, M. G. (2022). Evaluate sustainable human resource management in the manufacturing companies using an extended Pythagorean fuzzy SWARA-TOPSIS method. Journal of Cleaner Production. 370: 133380.

Samlı, R. ve Orman, Z. (2023). A comprehensive overview of web-based automated testing tools. İleri Mühendislik Çalışmaları ve Teknolojileri Dergisi. 4(1): 13-28.

Sampath, S. ve Sprenkle, S. (2016). Advances in Web Application Testing. Advances in Computers (pp. 155-191). New York: Elsevier Publishing.

Selenium.dev, (12.03.2023), Selenium Documentation. . (16.03.2025).

Selenium.dev, Selenium History. . (15.02.2025).

Shaheen, Z., Rauf, A., ve Hameed, B. (2013). Automated software testing: An insight into local industry. AWERProcedia Information Technolgy. Computer Science. 4: 408-415.

Sharma, M., ve Angmo, R. (2014). Web based automation testing and tools. International Journal of Computer Science and Information Technologies. 5(1): 908-912.

Sheremeta, O. (12.12.2024). Playwright Test Automation: Key Benefits and Features. . (27.03.2025).

Sinaga, A. M., Wibowo, P. A., Silalahi, A., ve Yolanda, N. (2018). Performance of automation testing tools for android applications. In 2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE) (ss. 534-539). IEEE.

Singh, D., Kumar, P. ve Kumar, S. (2014). Gap analysis between theory and practice of software testing methodologies in indian IT industry. International Journal of Computer Applications. 101: 25-33.

Stanković, M., Stević, Ž., Das, D. K., Subotić, M. ve Pamučar, D. (2020). A new fuzzy MARCOS method for road traffic risk analysis. Mathematics. 8(3): 457.

Stević, Ž., Puška, A., Brković, N. ve Tanackov, I. (2020). A Novel Integrated FUCOM-MARCOS Model for Evaluation of Human Resources in a Transport Company. Sustainability, 12(17), 6944.

Toledano, C. A. (2013, July). Web 2.0: the origin of the word that has changed the way we understand public relations. In Barcelona international PR conference, Barcelona, Spain.

Türkmen, M. ve Demirel, A. (2022). SWARA Ağırlıklı Bulanık COPRAS Yöntemi ile Tedarikçi Seçimi. Alanya Akademik Bakış, 6(1): 1739-1756.

Ulutaş, A., Karabasevic, D., Popovic, G., Stanujkic, D., Nguyen, P. T. ve Karaköy, Ç. (2020). Development of a novel integrated CCSD-ITARA-MARCOS decision-making approach for stackers selection in a logistics system. Mathematics. 8(10): 1672.

Umar, M. A., ve Zhanfang, C. (2019). A study of automated software testing: Automation tools and frameworks. International Journal of Computer Science Engineering (IJCSE). 6: 217-225.

Vaidya, N. (20.03.2025). edureka, What are the various Components of Selenium Suite?. . (20.04.2025).

Van Dijck, J., Poell, T., ve De Waal, M. (2018). The platform Society as a Contested Consept. The platform society: Public values in a connective world (pp.7-26). Oxford university press.

Venters, C. C., Capilla, R., Nakagawa, E. Y., Betz, S., Penzenstadler, B., Crick, T., ve Brooks, I. (2023). Sustainable software engineering: Reflections on advances in research and practice. Information and Software Technology. 164: 107-316

Wang, Y., Li, Y. ve Zhao, H. (2022). Sustainable supplier selection in Industry 4.0 using OPA-integrated fuzzy MARCOS method. Journal of Cleaner Production. 10: 23-51.

Yacan, İ. (2024). Çok Kriterli Karar Vermede Yeni Bir Yöntem Önerisi: Geliştirilmiş Kriter Ağırlıklandırma Yöntemi, Kesin Ve Bulanık Sayılarla Uygulamalar. Denizli: Pamukkale Üniversitesi Sosyal Bilimler Enstitüsü.

Yaralıoğlu, K. ve Karagöz, E. (2024). Çok Kriterli Karar Verme Yöntemleri (Algoritme Ve Yazılım). Ankara: Nobel Yayıncılık.

Yasar, K. (2024). Techtarget, What is web application (web apps) and its benefits. ,(25.01.2025).

Yener, H. A., Baştürk, F. ve Meçik, B. (2019). Yazılım Geliştirme ve Test Otomasyon ile Verimlilik Artışı: General Mobile. Karamanoğlu Mehmetbey Üniversitesi Mühendislik ve Doğa Bilimleri Dergisi. 1(1): 172-196.

Zavadskas, E. K., ve Podvezko, V. (2016). Integrated determination of objective criteria weights in MCDM. International Journal of Information Technology & Decision Making. 15(02): 267-283.

Zolfani, S. H., Görçün, Ö. F. ve Küçükönder, H. (2021). Evaluating logistics villages in Turkey using hybrid improved fuzzy SWARA (IMF SWARA) and fuzzy MABAC techniques. Technological and Economic Development of Economy. 27(6): 1582-1612.

Özcan, A. ve Ömürbek, N. (2020). Bir Demir Çelik İşletmesinin Performansının Çok Kriterli Karar Verme Yöntemleri İle Değerlendirilmesi. IBAD Sosyal Bilimler Dergisi. (8): 77-98.

Özdağoğlu, A., Keleş, M. K. ve Işıldak, B. (2021). Bulanık SWARA ve Bulanık MARCOS Yöntemleriyle Sivil Havacılıkta Kabin Memuru Seçimi. Gümüşhane Üniversitesi Sosyal Bilimler Dergisi. 12(2): 284-302.

Şengül, D., ve Cagıl, G. (2020). Bulanık swara ve bulanık analitik hiyerarşi prosesi yöntemi ile iş değerlemesi. Dicle Üniversitesi Mühendislik Fakültesi Mühendislik Dergisi. 11(3): 965-976.