T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ÇEVİK YAZILIM TEST SÜREÇLERİNDE RİSK ANALİZİ ÇALIŞMASI

YÜKSEK LİSANS TEZİ

Işılay PAMUK CANDAN

Endüstri Mühendisliği Anabilim Dalı

Mühendislik Yönetimi Bilim Dalı

HAZİRAN 2023

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ÇEVİK YAZILIM TEST SÜREÇLERİNDE RİSK ANALİZİ ÇALIŞMASI

YÜKSEK LİSANS TEZİ

Işılay PAMUK CANDAN

Endüstri Mühendisliği Anabilim Dalı

Mühendislik Yönetimi Bilim Dalı

Tez Danışmanı: Doç. Dr. Tülay KORKUSUZ POLAT

HAZİRAN 2023

Işılay PAMUK CANDAN tarafından hazırlanan "Çevik Yazılım Test Süreçlerinde Risk Analizi Çalışması" adlı tez çalışması 19.06.2023 tarihinde aşağıdaki jüri tarafından oy birliği ile Sakarya Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalı Mühendislik Yönetimi Bilim Dalı'nda Yüksek Lisans tezi olarak kabul edilmiştir.

Tez Jürisi

Jüri Başkanı :	**Doç. Dr. Esra TEKEZ** Sakarya Üniversitesi	.....
Jüri Üyesi :	**Doç. Dr. Tülay KORKUSUZ POLAT** Sakarya Üniversitesi	.....
Jüri Üyesi :	**Doç. Dr. Didem GÜLERYÜZ** Bayburt Üniversitesi	.....

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Sakarya Üniversitesi Fen Bilimleri Enstitüsü Lisansüstü Eğitim-Öğretim Yönetmeliğine ve Yükseköğretim Kurumları Bilimsel Araştırma ve Yayın Etiği Yönergesine uygun olarak hazırlamış olduğum "Çevik Yazılım Test Süreçlerinde Risk Analizi Çalışması" başlıklı tezin bana ait, özgün bir çalışma olduğunu; çalışmamın tüm aşamalarında yukarıda belirtilen yönetmelik ve yönergeye uygun davrandığımı, tezin içerdiği yenilik ve sonuçları başka bir yerden almadığımı, tezde kullandığım eserleri usulüne göre kaynak olarak gösterdiğimi, bu tezi başka bir bilim kuruluna akademik amaç ve unvan almak amacıyla vermediğimi ve 20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince Sakarya Üniversitesi'nin abonesi olduğu intihal yazılım programı kullanılarak Enstitü tarafından belirlenmiş ölçütlere uygun rapor alındığını, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun ortaya çıkması halinde doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi beyan ederim.

(19/06/2023).

(imza)

Işılay PAMUK CANDAN

Abime

TEŞEKKÜR

Tez çalışmalarının bütün aşamalarında yardımlarını esirgemeyen tez danışmanım değerli hocam Doç. Dr. Tülay KORKUSUZ POLAT'a en içten teşekkürlerimi sunarım. Ayrıca tüm hayatım boyunca maddi manevi desteklerini esirgemeyen çok kıymetli canım aileme ve eşime sevgilerimi sunarım.

Işılay PAMUK CANDAN

İÇİNDEKİLER

Sayfa

KISALTMALAR

AHP	:Analytic Hierarchy Process (Analitik Hiyerarşi Prosesi)
BWM	:Best–Worst Method (En İyi – Kötü Yöntemi)
ÇKKV	:Çok Kriterli Karar Verme
FMEA	:Failure Mode Effect Analysis (Hata Türü ve Etkileri Analizi)
HAZOP	:Hazard ve Operability (Proses Tehlike Analizi Metodu)
MAUA	:Multi-Featured Utility Analysis (Çok Nitelikli Yardımcı Program Analizi)
MCDM	:Multi Criteria Decision Method (Çok Kriterli Karar Verme Modeli)
SDLC	:Software Development Life Cycle (Yazılım Geliştirme Yaşam Döngüsü)
SWARA	:Step-Wise Weight Assessment Ratio Analysis (Kademeli Ağırlık Değerlendirme Oran Analizi)
TOPSIS	:Technique for Order Preference by Similarity to Ideal Solution (İdeal Çözüme Benzerliğe Göre Sipariş Tercihi Tekniği)
VIKOR	:Multi-Criteria Optimization ve Compromise Solution (Çok Kriterli Optimizasyon ve Uzlaşık Çözüm)
WASPAS	:Bütünleşik Ağırlıklı Toplam ve Çarpım Yöntemi
WPM	:Weighted Product Model (Ağırlıklı Çarpım Modeli)
WSM	:Weighted Sum Method (Ağırlıklı Toplam Modeli)

SİMGELER

$\mathbf{k_j}$	: j. Kriterin Katsayı
$\mathbf{\tilde{q}_j}$	: j. Kriterin Önem Vektörü
$\mathbf{\tilde{w}_j}$	: j. Kriterin Ağırlık Değeri
$\mathbf{w_j}$	: j. Kriterin Durulaştırılmış Ağırlık Değeri
O	: Hatanın Olasılık Değeri
F	: Hatanın Farkedilebilirlik Değeri
$\mathbf{E_1}$	: Zaman Bileşinin Etki Değeri
$\mathbf{E_2}$	: Kalite Bileşinin Etki Değeri
$\mathbf{E_3}$	: Maliyet Bileşinin Etki Değeri
$\mathbf{E_4}$	: Çözülebilirlik Bileşinin Etki Değeri
$\mathbf{E_5}$	: Kullanılabilirlik Bileşinin Etki Değeri

TABLO LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

ÇEVİK YAZILIM TEST SÜREÇLERİNDE RİSK ANALİZİ ÇALIŞMASI

ÖZET

Günümüzde gelişen teknoloji ile yeni rekabet ortamları oluşmuştur. Şirketlerin bu rekabet ortamında müşteri isteklerine hızlı geri dönüş yapması önem kazanmıştır. Şirketlerin müşteri isteklerini karşılayabilmek için yazılım uygulamaları kullanımı ihtiyacı ortaya çıkmıştır. Yazılım uygulamaları, yazılım geliştirme yaşam döngüleri (Software Development Life Cycle-SDLC) boyunca insan faaliyetleri ile geliştirildikleri için risklerin oluşması sıklıkla yaşanabilmektedir. Bir yazılımın güvenilir ve kaliteli olması, o yazılım ürünün hatasız veya en az hata ile çalışmasıyla mümkün kılınmaktadır. Bu nedenle, risklerin önüne geçilebilmesi için yazılım testi büyük rol oynamaktadır.

Bu çalışmada, Ankara'da yer alan yazılım sektöründe faaliyet gösteren bir savunma sanayi şirketinin haberleşme projesindeki yazılım test süreçlerinde ortaya çıkan riskler için risk analizi gerçekleştirilmiştir. Şirket, müşteri isteklerini karşılayan, en az hata ile yüksek kalitede çalışan bir uygulama geliştirmeyi amaçlamaktadır. Yürütülen çevik (agile) yazılım projesinde, tekrarlanan hatalar nedeniyle yazılım testleri için uzun süren zamanlar ve insan gücü olarak büyük bir efor harcanmaktadır. Projenin zamanında ve belirlenen bütçe çerçevesinde tamamlanabilmesi için yazılım test süreçlerinde ortaya çıkan risk faktörlerinin önceliklendirilmesi gerekmektedir. Yazılım testinde ortaya çıkan her hatanın projeye olan etkileri birbirinden farkı olacağı için etki alt bileşenlerinin belirlenmesi amaçlanmıştır.

Bu kapsamda proje rol almakta olan uzman kişiler ile görüşülmüştür. Yapılan görüşmeler neticesinde belirlenen etki alt bileşenlerinin ağırlıklarının belirlenmesinde Çok Kriterli Karar Verme tekniklerinden çalışmaya uygun olan bulanık SWARA tekniği kullanılmıştır. Yazılım testinde ortaya çıkan risk faktörleri arasındaki ilişkinin önceliklendirilmesi için risk yönetimi tekniklerinden FMEA kullanılmıştır. İlk olarak klasik FMEA'daki "olasılık, farkedilebilirlik, etki" olarak ifade edilen üç bileşenin çarpımı ile risk öncelik numara değeri bulunmuştur. Bu çalışmada, Klasik FMEA'nın üç bileşenin (olasılık, etki, farkedilebilirlik) çarpımı ile bulunun risk öncelik numarası değeri, Bulanık SWARA yöntemi ile bulunan etki alt bileşenlerinin ağırlıklarının entegre edilmesi ile literatürde rastlanmayan özgün bir "Ağırlıklandırılmış FMEA" modeliyle hesaplanmıştır.

Alt etki bileşenlerinin ağırlıkları kullanılarak hesaplanan risk öncelik numaralarına göre önceliklendirilen yazılım test süreçlerinde ortaya çıkan riskler için hafifletici ve önleyici planlar sunulmuştur.

RISK ANALYSIS STUDY IN AGILE SOFTWARE TEST PROCESSES

SUMMARY

Nowadays, the software sector, which is one of the important sectors with the advancing technology, is trying to take place in the ecosystem where competition is intense because it operates in a wide area. In the rapidly changing world with the development of technology, companies try to respond quickly to the requests of their customers. Software applications play an important role in responding to customer requests. Regardless of the sector, many companies use software applications both in their internal processes ve in their interactions with their customers. In this context, many projects are being prepared. Software Development Life Cycle (SDLC) stveards should be applied in order to develop the project with minimum bug. Throughout SDLC, bugs can often occur as all phases are developed by human activities. The reliability ve quality of a software is made possible by the fact that that software product works regular or with the least amount of bugs. Therefore, software testing plays a major role in avoiding risks.

SDLC is a development approach used in agile software development processes Software development lifecycle (SDLC) is a cost-effective and time-efficient process used to design high-quality software. The purpose of SDLC is to minimize project risks with forward planning to meet customer expectations during and after production. This lifecycle consists of small plans that allow the software development process to be tracked and measured.

Managing software development can be challenging due to changing requirements, technology upgrades and cross-functional collaboration. The software development lifecycle (SDLC) methodology provides a systematic management framework with specific outputs at each stage of the software development process. All stakeholders ensure that plans are made on software development goals and requirements, and that communication is strong with people at all levels during the development phase.

Software development life cycle (SDLC) is a cycle which being of planning, analysis, design, implementation, testing and maintenance. It is aimed to produce software that works in line with changing requirements by communicating with the customer at every stage throughout this cycle.

Software development lifecycle models can be showned as spiral model, agile software development, waterfall model, rapid application development model, incremental model, V-Process model. In the study, the agile software development model, which is one of the most widely used models today, is discussed.

In agile software projects, software testing is usually carried out in parallel with software development activities. When planning software development steps, it is planned as a 3-week development program. The requirements that will be developed within 3 weeks are opened to the software team as a feature in the digital environment.

Usually at the end of each sprint, a major release is received and testing of this improved minor software system is performed. In addition, test activities are carried out by obtaining an intermediate version without waiting for the end of the sprint. Before starting the software test, test scenarios containing the requirements are written, test plans are made and the test environment is prepared. Test procedures including test scenarios and test planning for test tools are made. Testing activities are performed when this test planning is ready when the main release arrives at the end of the sprint. In the test activities, the features or bugs opened to the software developers in the digital environment are assigned to the test team by the software developers if they are finished. The testing team tests validation of these finished requirements. Bugs that occur as a result of all tests are opened as bugs in the same digital environment and assigned to the relevant software developer. Every found bug or improvement is tracked through this digital environment every time a new version is released. In this way, features such as which bug was opened when, when it was closed, how long it was resolved, and how many times the same situation was repeated can be reported through these digital tools.

Software testing activities play an important role in the quality and reliable operation of a software product. Along with software testing activities, a software system/product is developed with high performance that meets customer expectations, has been verified, and conforms to user behaviors. Many bugs that are not noticed at the code stage are detected at an early stage with the intermediate version tests conducted within the sprint. Detecting the bug at an early stage eliminates irreversible problems.

The software testing process is the testing activities carried out to ensure that a software works with quality, reliability and minimum bugs. During the software testing process, the testing techniques/levels to be applied and the software/hardware tools required for testing are determined. Verification methods are determined. It is determined at which level the test scenarios should be written. Test procedures template and test scripts are prepared. The test environment is set up. It is verified whether the software works correctly. The results realized during the test phase are compared with the expected results. Observed bugs are recorded and the test result is reported.

Software tests are carried out automatically and manually depending on the structure of the software. Automation; It is a test done by writing mini code scripts and automated tools. Manual testing; means that each test step is run one by one with human power. Therefore, manual tests take longer than automation. Running all tests from scratch is costly and time-consuming activities.

In order to produce a reliable and high-quality software product, the application and testing phases of the software development lifecycle are required to be carried forward in parallel. In this way, it is verified whether the new features included in the resilient software affect the working parts. At this point, project managers will be able to recognize the relevant risks earlier and take the necessary measures. Testing is an important stage in the software development lifecycle and plays a decisive role in ensuring the reliability and quality of the software. Therefore, in order to improve test efficiency, it is important to determine and eliminate the risk parameters that will affect the test phase in advance.

In this study, a risk analysis was carried out for the risks that arise in the software testing processes in the communication project of a defense industry company operating in the software sector. The company aims to develop a high-quality application that meets customer requests ve runs with minimal bugs. In the agile software project carried out, long time ve great effort are spent for software tests due to repeated bugs. In order for the project to be completed on time ve within the determined budget, the risk factors that arise during the software testing processes should be prioritized. It is aimed to determine the sub-components of the impact, since the effects of each bugs that occurs in the software test will be different from each other.

In this context, experts who were involved in the project were interviewed. The Fuzzy SWARA technique, which is one of the Multi-Criteria Decision Making techniques, was used to determine the weights of the impact sub-components determined as a result of the interviews. FMEA, one of the risk management techniques, was used to prioritize the relationship between the risk factors that emerged in the software test. First, the ris priority number value was found by multiplying the three components expressed as "probability, detectability, ve severity" in classical FMEA. In this study, the risk priority number value, which is found by multiplying the three components of Classic FMEA (probability, severity, detectability), was calculated with a unique "Weighted FMEA" model, which is not found in the literature, by integrating the weights of the severity sub-components found with the Fuzzy SWARA method.

While doing this calculation, the severity of risks on the project were identified as five sub-severity components. The weights of five subcomponents were found with Fuzzy SWARA. Then, the formula of classical FMEA is used. However, here, the severity value was determined for each identified risk and the weights of sub-severity components with this severity value was multiplied.

In the study, after the risks were determined, classical FMEA calculation was made for the risk prioritization study. Due to the deficiencies of the classical FMEA technique, such as having the same RPN values and the weights of the FMEA components, the weighted FMEA model was created. RPN value was calculated with the new model using the same risk items.

Weighted FMEA results with classical FMEA were compared. According to the results of the comparison, it was observed that the risk priority rankings of the other items, except for the first two items, were different from each other.

Mitigating ve preventive plans are presented for the risks that arise in software testing processes that are prioritized according to the risk priority numbers calculated using the weights of the sub-impact components.

1. GİRİŞ

Günümüzde farklı sektörlerde faaliyet gösteren birçok firma, rekabetin yoğun olduğu ekosistemde yer almaya çalışmaktadır. Teknolojinin gelişmesiyle hızla değişen dünyada şirketler, müşterilerinin isteklerine hızlı bir şekilde cevap vermeye çalışmaktadır. Müşteri isteklerine cevap vermede yazılım uygulamaları önemli rol oynamaktadır. Sektör ayrımı olmaksızın pek çok işletme hem kendi iç süreçlerini yürütürken hem de müşterileri ile olan etkileşimlerinde yazılım uygulamaları kullanmaktadırlar. Yazılım uygulamaları yazılım geliştirme yaşam döngüleri (Software Development Life Cycle-SDLC) boyunca insan faaliyetleri ile geliştirildikleri için hataların oluşması sıklıkla yaşanabilmektedir (Singh ve ark, 2023).

SDLC müşteri beklentilerini karşılamak için yüksek kaliteli yazılımlar tasarlamak, geliştirmek ve test etme süreci olarak tanımlanabilir (Nath ve ark, 2023). Yazılım geliştirme yaşam döngüsü: gereksinim, tasarım, uygulama, test ve kurulum/kontrol aşamalarından oluşmaktadır (Lee ve ark 2020). Yazılım geliştirme yaşam döngüsü boyunca söz konusu yazılımın çalışması esnasında herhangi bir hatanın tetiklenmesi ile yazılım hataları oluşabilir ve hatalar zamanında tespit edilip giderilmezse sorunlar ve maliyetler artabilir (Lee ve ark, 2020). Yazılım geliştirme projeleri kapsamında, proje yöneticileri yazılımın sorunsuz çalışmasını sağlamak için pek çok risk ile uğraşmak zorundadırlar (Pinciroli ve ark, 2022).

Güvenilir ve kaliteli bir yazılım ürünü ortaya çıkarmak için "yazılım geliştirme yaşam döngüsü" nün uygulama ve test aşamaları paralel olarak gerçekleştirilmelidir. Bu şekilde proje yöneticileri ilgili riskleri daha erken zamanlarda fark edip gerekli önlemleri alabileceklerdir. Test çalışmaları, yazılım geliştirme yaşam döngüsünde önemli bir aşamadır ve yazılımın güvenilirliğini ve kalitesini sağlama konusunda belirleyici rol oynamaktadır (Singh ve ark, 2023). Bu nedenle test verimliliğini iyileştirmek için test aşamasını etkileyecek parametreleri-riskleri önceden belirleyerek ortadan kaldırmak önem taşımaktadır.

Geliştirilen yazılımın kalitesi hatasız çalışmasına bağlıdır (Rhmann ve ark, 2020). Hatalar SDLC nin herhangi bir aşamasında ortaya çıksa da bunların erken teşhisi,

hataların oluşmadan fark edilmesi ve gerekli önlemlerin alınması önemlidir. Bu noktada SDLC de risk yönetimi çalışmalarının gerekliliği öne çıkmaktadır.

Yazılım geliştirme yaşam döngüsünün tüm aşamalarında donanım, yazılım, teknoloji ve insanın yer almasından dolayı, tüm aşamalar potansiyel risk kaynağı olarak görülebilir (Kumar ve Yadav, 2015). Bu nedenle yazılım geliştirme yaşam döngüsü için risk yönetimi oldukça önemlidir. Etkili bir risk yönetimi ile yazılım projesinin istenilen verimliliğe ve kaliteye sahip olması ve istenilen zamanda/bütçede tamamlanması sağlanabilmektedir (Kumar ve Yazav, 2015). Yazılım geliştirme yaşam döngüsü içerisinde risklerin belirlenerek gerekli önlemlerin alınması için en uygun aşama, kontrollerin yapıldığı test aşamasıdır (Singh ve ark, 2023). Yazılım geliştirme yaşam döngüsü içinde test aşaması en çok zaman alan ve maliyet gerektiren aşamalardan biridir. Test aşamasında; yazılım projesinin gereksinimlerine göre test planı oluşturulmalı, test araçları belirlenmeli ve test prosedürü oluşturulmadır. Test prosedürüne göre uygulama esnasında gerçekleştirilen testler ile riskler bulunmaktadır. Yazılım ekiplerinin ilgili riskleri düzelterek hazırladıkları yeni sürümü test ekibine vermesi ile bu adımlar riskler tamamen ortadan kaldırılıncaya kadar devam etmektedir. Bir önceki sürümde tespit edilen riskin yeni sürümde de var olması test aşamasının uzun sürmesine neden olmaktadır. Riskleri ortadan kaldıracak faaliyetleri belirlerken hangi riskin daha öncelikli olduğunun belirlenmesi için risk önceliklendirme yapılmalıdır. Risk önceliklendirmede söz konusu risklerin yazılım projesine olan farklı etkilerinin göz önüne alınmamasından dolayı projenin önemli konularına daha az etki eden riskler önceliklendirilebilir. Bu da yine test aşamasının uzamasına neden olabilmektedir. Riskler kullanılan risk değerlendirme metodolojisinin bileşenlerine göre önceliklendirilmektedir. Etki bileşeni, risk büyüklüğünü belirlemede pek çok risk değerlendirme tekniğinde kullanılan önemli bir bileşendir (Souza ve ark, 2021). Ancak risklerin projeye olan etkileri hep aynı şekilde olmayabilir. Klasik risk değerlendirme metodolojilerinin dezavantajlı yönlerinden birisi de etki bileşenini hep aynı şekilde değerlendirmeleridir. Bu çalışmada, olası proje yazılım risklerini değerlendirirken klasik hata türü ve etkileri analizinin bileşenlerinden biri olan etki bileşeninin: zamana etki, kaliteye etki, maliyete etki, çözülebilirliğe etki ve kullanılabilirliğe etki olmak üzere beş alt etki bileşenine

ayrıldığı yeni bir risk değerlendirme yaklaşımı önerilmektedir. Yeni yaklaşım ile klasik risk değerlendirme metodolojilerinin tüm risklerin etkilerini aynı şekilde değerlendirmesi dezavantajının önüne geçmek amaçlanmaktadır. Ayrıca önerilen modelde kullanılan alt etki bileşenlerinin yazılım projesi üzerindeki etkisinin aynı olamayacağından dolayı, alt etki bileşenleri çok kriterli karar verme tekniklerinden Bulanık SWARA (Step-wise Weight Assessment Ratio Analysis) yöntemi ile ağırlıklandırılmıştır. Yapılan çalışma hem farklı etki türlerinin proje üzerindeki etkilerinin değerlendirilmesini sağlaması hem de yazılım geliştirme süreci risklerinin değerlendirilmesi için yeni bir FMEA modeli önerilmiştir.

2. KAYNAK ARAŞTIRMASI

2.1. Çok Kriterli Karar Verme ve Risk Yönetimi Teknikleri

Çevik (Agile) yazılım geliştirme yaşam döngüsü süreci, müşteri odaklı olarak ilerlemekte ve işbirlikçi geliştirme gerektirmektedir (Moyano ve ark, 2022). Geliştirme sürecinde müşteri tarafından istenilen istekler yazılım süreçlerinde birtakım değişikliklere neden olmaktadır. Kaliteli yazılımlar, kabul edilebilir düzeyde hatasız, planlanan proje süresi ve maliyetinde bitirilip, beklentileri karşılayabilen ve sürdürülebilir özelliklere sahip yazılımlardır (Ouriques ve ark, 2023). Bu noktada yazlım test süreçleri, müşterilere kaliteli ürün sağlama konusunda önemli rol oynamaktadır. Bu süreçte sık değişen gereksimlerin karşılanması için yazılım testinin sürekli yapılması gerekmektedir (Ouriques ve ark, 2023). Yazılım testinde sürekli tekrar eden hataların birçok kriter nedeniyle önceliklendirilmemesi ve çözümlenememesi sonucu; projede maliyet, zaman, kalite vb. açıdan riskler oluşabilmektedir. Literatürde önceliklendirme, sıralama, ağırlıklandırma, yönetimi konularında birçok çalışmaya yer verilmiştir.

Karar verilirken çok fazla kriterin değerlendirilmesi gereken durumlar için literatürde Çok Kriterli Karar Verme (MCDM) tekniklerinin kullanıldığı pek çok çalışma bulunmaktadır. Sangwan ve Dhvea (2022) yaptıkları çalışmalarında bulut ağ hizmetleri seçimi ve sıralama için MCDM tekniklerinden olan AHP (Analytic Hierarchy Process) ve TOPSIS (Technique for Order Preference by Similarity to Ideal Solutions)' in entegre ederek uygulamışlardır. Nazim ve arkadaşları (2022), yazılım gereksimlerini önceliklendirmek ve sıralamak için AHP ve TOPSIS yöntemlerini entegre şekilde kullanmışlardır. Ranjbar ve arkadaşları (2022), proje potföy seçme ve çizelgeleme çalışması için bulanık AHP ve TOPSIS yöntemi uygulamışlardır. Singh ve arkadaşları (2023), yazılım test süreçleri parametrelerini önceliklendirme ve sıralamak için bulanık AHP ve TOPSIS yöntemini entegre ettikleri bir çalışma yapmışlardır.

Adalı ve Işık (2017) tedarikçi seçimi probleminde seçim kriterlerinin ağırlıklarının bulunmasında SWARA (Step-wise Weight Assessment Ratio Analysis) yöntemini ve en iyi seçimi yapmak için WASPAS (Weighted Aggregated Sum Product Assessment) yöntemini kullanmışlardır. Kersuliene ve arkadaşları (2010) yasama sistemlerindeki yöntemlerinin çok özellikli (ekonomik, sosyal vb.) olarak değerlendirilebilmesi için SWARA yöntemi kullanılmıştır. Zolfani ve arkadaşları (2015) karar ve politika organlarında en üst düzeydeki karar vericilerin, AR&GE projeleri seçiminde SWARA yöntemini ele alınmıştır. Tablo 2.1'de MCDM tekniklerinin kullanıldığı farklı uygulama örnekleri gösterilmektedir.

Tablo 2.1. Çok kriterli karar verme tekniklerinin uygulandığı çalışmalar.

Yazar(lar)	**Çalışma Amacı**	**Kullanılan Teknikler**
Alimardani ve ark. (2013)	Çevik Ortamda Tedarikçi Seçimi	SWARA, VIKOR
Dhvea ve Sangwan (2022)	Bulut Servisleri	AHP, TOPSIS
Naeemah ve Wong (2023)	Yalın Üretim Araçları Seçimi	TOPSIS, BWM
Soltan ve ark. (2023)	Endüstriyel Robot Seçimi	FAQT
Sarkodie ve ark. (2022)	Yenilenebilir Enerji Kaynakları Optimizasyonu	MOORA,TOPSIS,COPRAS
Seker (2022)	Sürdürülebilir Atık Yönetimi Değerlendirmesi	RFID, GIS, GPRS
Lohakare ve ark. (2022)	Malzeme Seçimi	AHP
Sarkar ve ark (2022)	Havza Önceliklendirmesi	AHP, TOPSIS, VIKOR
Chodha ve ark (2022)	Kaynak Robotu Seçimi	TOPSIS

Çok kriterli karar verme tekniklerinin kullanılması esnasında özellikle uzman görüşlerinden yararlanılan tekniklerde, uzmanların sübjektif olabilmesi, sonuçları kesinlikten uzaklaştırabilmektedir. Bu nedenle pek çok araştırmacı ÇKKV tekniklerinin bu tarz dezavantajlarını ortadan kaldırmak için, ÇKKV tekniklerini Yapay Zeka tekniklerinden Bulanık Mantık tekniği ile entegre şekilde kullanmışlardır. Zorlu ve Dede (2023) buzul çevresi yer şekillerinin jeomiras potansiyellerini değerlendirmek için yerleşim kriterli ağırlıklandırma çalışmasında bulanık SWARA yöntemi kullanmıştır. Şengül ve Çağıl (2020) doğru işte doğru ücret politikası çalışmasında iş değerleme kriterlerinin ücret yönetimi ile entegre edilmesinde kriterlerin belirlemesi ve karşılaştırılmasını bulanık SWARA ve AHP yöntemi ile

yapmışlardır. Korkusuz Polat ve Kara (2021) çalışmalarında personel seçimi için Bulanık DEMATEL (The Decision Making Trial ve Evaluation Laboratory) ve Bulanık VIKOR (Višekriterijumsko Kompromisno Rangiranje) yöntemini kullandıkları entegre bir model sunmuşlardır. MCDM tekniklerinin bulanık olarak kullanıldığı uygulama örnekleri Tablo 2.2'de gösterilmektedir.

Tablo 2.2. Bulanık çok kriterli karar verme tekniklerinin uygulandığı çalışmalar.

Yazar(lar)	**Çalışma Amacı**	**Kullanılan Teknikler**
Ansari ve ark. *(*2020)	Tedarik Zinciri Yönetimi	Fuzzy SWARA, COPRAS
Ranjbar ve ark. (2022)	Proje Yönetimi	FAHP, TOPSIS
Singh ve ark. (2023)	Yazılım Test Yönetimi	FAHP, Fuzzy TOPSIS
Abdullah ve ark. (2023)	Nükleer Santral Tesis Seçimi	FAHP
Rabia ve Bellabdaoui, (2023)	Yük Taşımacılığında Simülasyon Çalışması	IF-AHP
Abusaeed ve ark. (2023)	Maliyet Faktörlerinin Önceliklendirilmesi	FAHP
Abdullah ve ark. (2023)	Sürdürülebilir Üretim Uygulaması	FAHP, Fuzzy TOPSIS
Esmaelnezhad ve ark. (2023)	Stratejik İttifakların Seçimi	Fuzzy SWARA
Keshteli ve ark. (2023)	Gıda Sektöründe Yeşil Tedarikçi Seçimi	Fuzzy TOPSIS
Abdul ve ark. 2023	Girişimcilik Engellerinin Önceliklendirilmesi	FAHP

Bir çok teknoloji ile entegre çalışılması, yapısal/kavramsal modellerin ve kullanılan standartların farklı olması gibi nedenlerden dolayı yazılım faaliyetlerinde riskler oldukça yaygındır. Yapılan işin performansı üzerinde istenmeyen sonuçlar doğurabilecek riskleri değerlendirerek hataların meydana gelmesini veya tekrarlanmasını engellemek için kullanılan pek çok teknik vardır: Karar Matrisi Risk Analizi Yöntemi (DMRA), HAZOP (Hazard ve Operability), Hata Ağacı Analizi (FTA), Olay Ağacı Analizi (ETA), Hata Türü ve Etkileri Analizi (FMEA) gibi. Bu teknikler, problemlerin temel nedenini analiz etmek için kullanılırken, tekniklerin uygulanabilirliklerini ve etkinliklerini arttırarak riskleri daha etkin yönetebilmek için genellikle bulanık küme teorisi, sosyal ağ analizi, yorumlayıcı yapısal modelleme (interpretive structural modeling) ve monte carlo simülasyonu gibi tekniklerin yanı

sıra çok kriterli karar verme teknikleri ile birlikte kullanılmaktadırlar (Tomak ve Korkusuz Polat, 2022). Kullanılan çok kriterli karar verme yöntemlerinin işlevi genellikle riskleri önceliklendirmektir. Sosyal hayattaki sorunları ele alarak risk yönetimi teknikleri kullanan Gökler ve arkadaşları (2022) tarafından çalışılan bir araştırmada huzurevlerindeki ortam koşullarına ait risklerin belirlenmesi ve çözülmesi için Fine-Kinney ve ANFIS yöntemleri kullanılmıştır. Taheriyoun ve Moradinejad (2015) tarafından atık su analizinde tehlikeli maddelerin belirlenmesi ve tehlikenin giderilmesi için uygulanacak yöntemin seçiminde FTA (Failure Tree Analysis) ve Monte Carlo simülasyonu kullanmışlardır. Tablo 2.3'de farklı yöntemlerle yapılan risk değerlendirme çalışma örnekleri gösterilmektedir.

Tablo 2.3. Risk değerlendirme çalışmaları.

Yazar(lar)	**Çalışma Amacı**	**Kullanılan Teknikler**
Verade ve ark. (2022)	Ürün Geliştirme Risk Analizi	FMEA
Souza ve ark. (2021)	Covid 19 Projelerinde Risk Yönetimi	FMEA
Akyuz ve Celik (2018)	Denizcilik Risk Değerlendirmesi	FMEA, IT2FSs
Chen ve ark. *(*2021)	Süspansiyon Arızasının Risk Analizi	FMEA
Shafiee ve ark. (2022)	Deniz Altı Tesislerinde Risk Analizi	FMEA
Takahashi ve ark. (2022)	Yazılım ve Donanım Testleri Risk Değerlendirmesi	HAZOP
Wang ve Gao (2012)	Bilgi Veritabanı Oluşturma Risk Değerlendirmesi	HAZOP
Shaw ve Blundell (2007)	Nükleer Atık Risk Değerlendirmesi	WASOP, HAZOP
Taheriyoun ve Moradinejad (2015)	Atık Su Risk Analizi	FTA, Monte Carlo
Xie ve ark. (2021)	Bulut Modeli Teori Bağlantısına Dayalı Risk Analizi	Risk Matrix
Zhang ve ark.(2017)	Deniz Altı Pens Kurulum İşlemleri Risk Analizi	Fuzzy Risk Matrix

Ayrıca literatürde MCDM teknikleri ve risk analizi yöntemlerinin entegre edildiği çalışmalarda bulunmaktadır. Yener ve Can (2021) tedarik zinciri yönetiminde sermaye ve zaman risklerini önlemek için bulanık SWARA ve FMEA (Failure Mode Effect Analysis) risk değerlendirme yöntemini birlikte kullanılmışlardır. Ünlükal ve Yücel (2021) havacılık sektöründe üretim süreçlerindeki risklerin önceliklendirmesi için bulanık TOPSIS ve FMEA yöntemlerini entegre etmişlerdir. Souza ve arkadaşları

(2021) karmaşık durumlarda risk önceliklendirme çalışmalarını iyileştirmek için FMEA ve çok kriterli karar verme yöntemleri kullanarak bir çalışma yapmışdır. Liu ve arkadaşları (2015) dizel motor sistemindeki hataları FMEA yöntemi ile belirleyerek hataları sıralamak için AHP ve VIKOR yöntemlerini kullanmışlardır. Pourmadadkar ve arkadaşları (2019), sağlık sektöründe kronik rahatsızlıklarda risk değerlendirmesini gerçekleştirmek ve kaliteyi arttırmak için VIKOR ve FMEA yöntemini ele almışlardır. Khalilzadeh ve diğerleri (2020) enerji sektöründe risklerin belirlenmesi ve sıralaması için yaptıkları çalışmada FMEA ve GRA (Grey Relationship Analysis)- VIKOR yöntemlerini entegre şekilde kullanmışlardır. Chen ve arkadaşları (2022) çevre kirliği konusunda kırmızı gelgit riskinin değerlendirmesinde AHP-TOPSIS CRITIC (inter criteria correlation)-ASSETS (assessment of estuarine trophic status) ve Monte Carlo yöntemlerini entegre şekilde kullandıkları bir uygulama yapmışlardır.

Shaw ve Blundell (2007), nükleer atıkların en aza indirilmesi için tehlikelerin belirlenmesi ve önleyici uygulamaların belirlenmesi için HAZOP (Hazard ve Operability Analysis) ve WASOP (Waste Ve Source material OPerability Study) yöntemlerini kullanarak çalışma yapmışlardır. Tablo 2.4'de ÇKKV tekniklerinin risk değerlendirme teknikleri ile birlikte kullanıldığı çalışmalara örnekler gösterilmektedir.

Tablo 2.4. Çok kriterli karar verme ve risk yönetemi tekniklerinin birlikte uygulandığı çalışmalar.

Yazar(lar)	**Çalışma Amacı**	**Kullanılan Teknikler**
Yu ve ark. (2023)	Denizaltı Boru Hattı Risk Analizi	FMEA, FAHP,TODIM
Koohathongsumrit ve Chankham (2023)	Tedarik Zinciri Rota Seçimi Risk Değerlendirme	FRAM, BWM, MARCOS
Marhavilas ve ark. (2020)	Petrol Tesisinde Risklerin Belirlenmesi	HAZOP, FAHP
Zhang ve Xu (2021)	Tıp ve Sağlık Alanında Risk Yönetimi	FMEA, Fuzzy Wıngs-G1
Alvve ve ark. (2021)	İnşaat Projelerinde Risk	SWARA, WASPAS, FMEA

Tablo 2.4. (Devamı) Çok kriterli karar verme ve risk yönetemi tekniklerinin birlikte uygulandığı çalışmalar.

Yazar(lar)	Çalışma Amacı	Kullanılan Teknikler
Ahsan ve ark. (2022)	Bulut modeli Teorisi ile Üretim Sürecinin Değerlendirilmesi	FMEA, TOPSIS, CMT
İlbahar ve ark. (2022)	Yenilenebilir Enerji Yatırımları için Risk Değerlendirme	FAHP, FMEA, prospect theory
Ok ve ark. (2022)	Radyolojik Kazaların Önceliklendirilmesi	Risk Matrix, AHP
Fattahi ve Khlilzadeh (2018)	Risk Analizi İyileştirmesi	Fuzzy FMEA, AHP, MULTIMOORA
Souza ve ark. (2021)	Covid 19 Projelerinde Risk Yönetimi	FMEA.CPP
Kumari ve ark. (2023)	Üretim Tesisi Atık Su Zararlarının Etkisinin Azaltılması	Fuzzy FMEA, AHP, TOPSIS

Literatüre bakıldığında, Durhan (2016) yapmış olduğu çalışmadaki gibi FMEA'nın üç bileşeninin ağırlıklandırılarak yeni FMEA modelinin oluşturulduğu uygulamalar vardır. Ancak, yapılan kaynak araştırması sonucunda, yazarın bilgisine göre yazılım testleri sonucu uygulamada çıkan hataların önceliklendirilmesi ve projeye olabilecek farklı etki boyutlarının alt bileşenlere ayırılarak farklı ağırlıklarla değerlendirildiği bir çalışmaya göre rastlanmamıştır. Bu çalışma hem yazılım test risklerinin değerlendirilmesi hem de riskin farklı etki boyutlarının da değerlendirilmeye katılmasını sağlaması açısında literatüre önemli katkılarda bulunmaktadır. Çalışmada risk değerlendirme yöntemi olarak hem klasik FMEA hem de farklı etki türlerine göre dataylandırılmış FMEA kullanılmaktadır. Farklı etki türlerinin ağırlıklarını belirlemek için Bulanık SWARA yöntemi kullanılmıştır.

Mevcut çalışmada yazılımda test süreçlerinde en çok tekrar eden hataların projeye olan etkilerine göre önceliklendirilmesi için ağırlıklandırılmış bir metadoloji uygulanmaktadır. Uygulanan teknikler alt bölümlerde açıklanmıştır.

3. MATERYAL VE YÖNTEMLER

3.1. Yazılım Proje Yönetimi

Proje yönetimi, proje hedeflerine ulaşmak için zaman, maliyet ve kalite fonksiyonları göz önünde tutularak, mühendislik aktivitelerinin doğru olarak planlanması, organize edilmesi, kontrol edilmesi ve denetlenmesi ile projeye liderlik edilmesidir (Çalışkan ve ark, 2021). Bilgi teknolojilerinin hızlı gelişimi ve müşteri isteklerindeki değişimler, mevcut proje içerisinde değişikliğe sebep olabilmekte ve önceki projeler uygulanırken kazanılmış deneyimler ise genellikle yeterli gelmemektedir. Yazılım proje yönetiminin hedefi kaynakların, risklerin, kapsamın ve maliyetin doğru yönetilmesini sağlayarak yazılım geliştirme faaliyetlerini disiplinli, iyi yönetilen, ürünlerin ya da sonuçların kaliteli teslimini güvence altına alarak, belirli bir zaman ve bütçe kısıtı altında gerçekleştirmektir (Calp ve Akcayol, 2015).

3.2. Yazılım Geliştirme Yaşam Döngüsü

Yazılım geliştirme yaşam döngüsü, müşteri beklentileri, kaliteyi, verimliliği sağlayan yüksek kaliteli yazılım sistemlerini geliştirmek için uygulanan sistamatik bir yaklaşımdır. Yazılım sisteminin üretiminden müşteriye kullanım süreci boyunca tüm aşamalara yazılım geliştirme yaşam döngüsü (Software Developmant Life Cycle (SDLC)) adı verilir (Yılmaz, 2007).

Yazılım projesi boyunca gereksinimler sürekli değiştiği ve kapsam genişlediği için yazılımın organize bir şekilde geliştirilmesi gerekmektedir. SDLC, birbirini takip eden döngü şeklinde adımlardan oluşur. Yazılım nasıl geliştirileceği, tasarlanacağı, iyileştirileceği ve sürdürüleceği açıklayan bir plan oluşur. SDLC'de her aşama bir önceki aşamanın çıktılarını girdi olarak kullanır.

3.2.1. Yazılım geliştirme yaşam döngüsü aşamaları

Yazılım geliştirme yaşam döngüsü altı aşamadan oluşmaktadır (Kılınç, 2006).

- Planlama:

Müşterin isteğini karşılamak için toplantılar yapılarak veri toplanır. Müşterinin ne istediği anlaşılmaya çalışılır. Ürünün amacının ne olduğu belirlenir ve buna göre ekip planlanır. İş analistleri tarafıdan gereksinimler oluşturulmaya başlanır.

- Analiz

Müşteri tarafındna iletilen istekler gereksinim haline getirilir. Gereksinimler detaylı şekilde incelenir. Geliştirilecek ürünün gereksinimlerini açıkça tanımlayan doküman oluşturulur. Tüm ekip ve paydaşlar ile paylaşılır.

- Tasarım

Açıkça ifade edilen gereksinimlere göre yazılımın mimari tasarımı oluşturulur. Yazılımı içeren bileşenler ve sistemin mantıksal, fiziksel ve veri yapısı oluşturulur. Tasarım doğrulama kriterleri belirlenir. Yazılım tasarım dokümanları oluşturulur.

- Uygulama

Yazılımın programlveığı ve kodun geliştirildiği aşamadır. Kod geliştirilirken hazırlanan tasarım dokümanınan ve belirlenen yazılım standartlarına ve yönergelerine uyulması gerekmektedir.

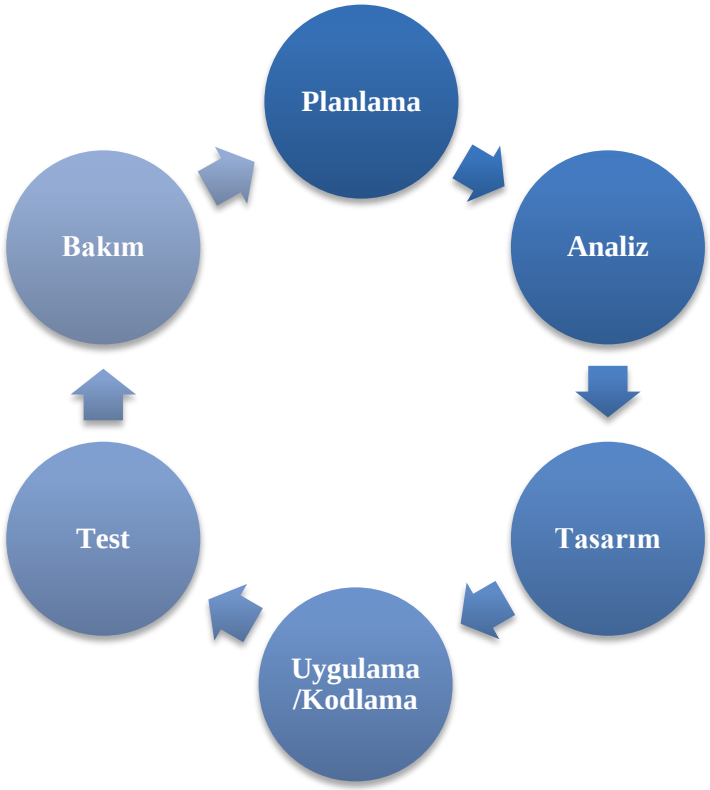
- Test

Kod oluşturulduktan sonra gereksinimlerin karşılandığını ve müşteri ihtiyaçlarının gerçekleştiğinin doğrulandığı ve onaylandığı aşamadır. Bu aşamada çeşitli test teknikleri uygulanır.

- Bakım

Yazılım sisteminin devreye alındığı kısımdır. Geliştirilen yazılım ürünü müşteriye teslim edildikden sonra karşılaşılan sorun ve çözülmesi istenen gereksinimleri içeririr. Bu sorun ve gereksinimlerin düzeltildiği ve iyileştirildiği aşamadır.

Şekil 3.1.'de yazılım geliştirme yaşam döngüsü gösterilmektedir.



Şekil 3.1. Yazılım geliştirme yaşam döngüsü (SDLC).

Yukarıda açıklanan aşamalar yazılım geliştirme sürecinde uygulanarak geliştirme sonucunda müşteriye kaliteli ve güvenilir bir ürünün teslim edilmesi için uyulması gereken bir döngüdür. Karmaşık ve çok büyük projelerde süreçleri daha küçük yapılara ayıran ve organize bir şekilde çalışmayı sağlayan geliştirme modelleri bulunmaktadır. Her model kendine özgü özelliklere sahiptir (Govil ve Sharma, 2022). Bu modelle aşağıda ifade edilmiştir.

1. Spiral Model
2. Çevik Yazılım Geliştirme
3. Şelale Modeli
4. Hızlı Uygulama Geliştirme Modeli (RAD)
5. Artımlı Model
6. V-Proses Modeli

Birçok yazılım geliştirme modeli olmasına rağmen günümüzde yazılım projelerinde en popüler olan Çevik model ele alınmıştır.

3.2.2. Çevik (agile) yazılım geliştirme modeli

Çevik yazılım geliştirme, Çevik Yazılım Geliştirme Manifestosu'nda yer alan yazılım geliştirmedeki farklı yaklaşımları ifade eden şemsiye bir terimdir (Moyano ve ark, 2022). Bu yaklaşımların odak noktası, değişime yanıt vermedeki esnekliktir. (Williams ve Cockburn, 2003). Bu yaklaşımlar ayrıca kendi kendini koordine eden, ekipler arası takım çalışmasına dayanan, yüksek kaliteli yazılım ürünlerinin en hızlı şekilde ortaya çıkmasını sağlayan, müşteri odaklı yazılım geliştirme metodunu sağlar.

Bu uygulamaların temel fikirleri, dört değerin tanımlandığı Çevik Yazılım Geliştirme Manifestosu'nda tanımlanan bu uygulamanın dört temel değeri şu şekildedir: (Ouriques ve ark, 2023)

- Bireyler ve etkileşimler, süreçler ve araçlardan daha önceliklidir.
- Çalışan yazılım, kapsamlı dokümantasyondan daha önceliklidir.
- Müşteri işbirliğine, sözleşme müzakeresinden daha fazla öncelik verilir.
- Değişime yanıt vermek, bir planı takip etmekten daha önceliklidir.

Çevik yazılım geliştirme methodolojisi ile ekipler ve paydaşlar süreç boyunca birlikte çalışır ve bu iletişim karşılıklı olarak kısa bildirimler alınmasını sağlar. Bu noktada yazılım sürecinde ortaya çıkabilecek hata oranını azaltılır. Süreç boyunca değişen gereksinimler projenin son aşamasında bile kabul edilebilebir ve yazılım buna göre değiştirilebilir.

1990'ların ikinci yarısında farklı yazılım dilleri, farklı lokasyonlar, farklı projeler birçok uygulayıcı tarafından paralel olarak kullanılmaya başlandı. Ülke bazlı farklı çevik yöntemler ve geliştiricileri Tablo 3.1'de gösterilmiştir (Kutluay, 2018).

Tablo 3.1. Çevik yöntemler.

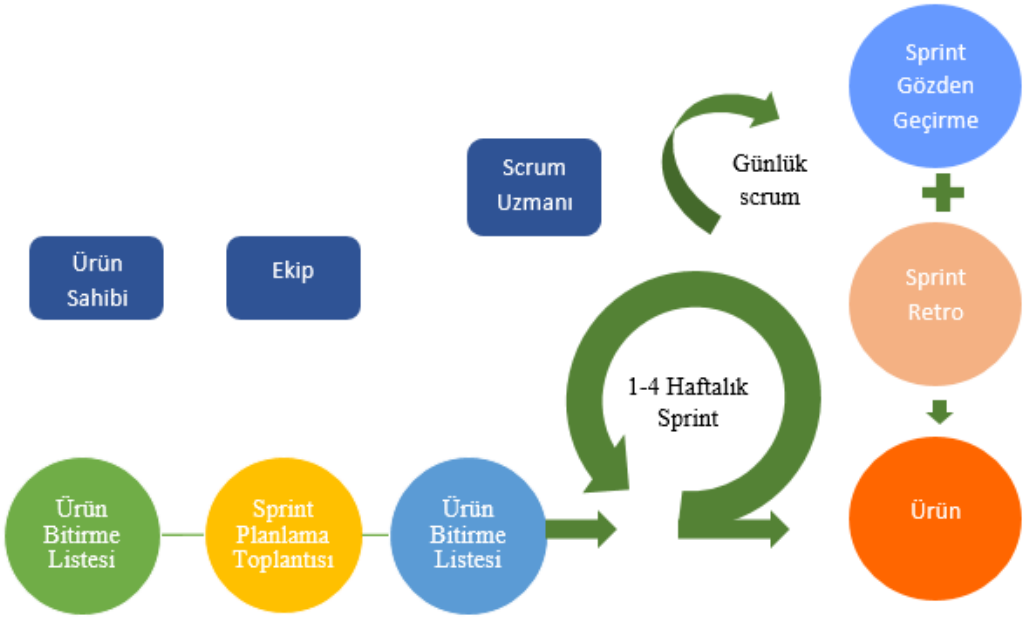
Ülke	Yöntem	Geliştirici
Amerika	eXtreme Programming	Ken Beck & Eric Gamma
	Scrum	Ken Schwaber & Jeff
	Crystal	Alistair Cockburn
	Adaptive software development	Jim Highsmith
	Lean software development	Tom & Mary Poppendieck
Avrupa	Dynamic systems development method	Dane Faulkner
Avustralya	Feature driven development	Peter Code & Jeff DeLuca

3.2.2.1. Scrum

Her zaman müşteri memnuniyetini ön planda tutarak yazılım faaliyetleri sürdürülen çevik yazılım geliştirme yöntemlerinde en çok kullanılan model scrum metodolojisidir. Scrum, başarılı bir ürün geliştirmek için müşteri ihtiyaçları doğrultusunda büyük bir projeyi daha küçük parçalara ayırır ve küçük planlamalar ile hedefe ulaşmayı sağlar (Butt, 2022).

Scrum süreci, yazılım projesi için gerekli olan gereksinimlerin önceliklendirilmiş bir listesi olan bir ürün biriktirme listesinin oluşturulmasıyla başlar. Ürün sahibi, ürün biriktirme listesini tanımlamaktan sorumludur. Proje, "sprint" olarak adlandırılan küçük kısımlara ayrılır ve 1-4 haftalık süreçler halinde gerçekleştirilir. Her sprintin başlangıcında ekip, ürün biriktirme listesinin hangi özelliğinin yapılacağını planlar ve zaman çizelgesi oluşturulur. Scrum ustaları ve ekip üyeleri, değişiklikleri iyileştirmek ve uyarlamak için önceki yinelemenin bitiminden sonra yeni yinelemeyi planlayabilir. Sprint biriktirme listesi, öncelikli özelliklerin bir listesidir ve ürünün tüm işlevlerini içerir. Ekip, ürün bitirme listesinde en yüksek inceliğe sahip özelliği/gereksinimi seçerek geliştirmeye başlar. Her gün günlük scrumlar yapılır ve proje ilerleyişi ile ilgili

güncel durum paylaşılır. Sprintlerde her hafta geliştirme durumuna göre ara sürümler alınabilir veya sprint sonu sürüm alınır ve testleri yapılır. Test sonuçları ekiple paylaşılır. Sprint sonunda, sprint gözden geçirmesi yapılır ve çıkan ürün değerlendirilir. Sprint retro toplantıları, ekibin kendini değerlendirmesi ve geliştirmesi için yapılan toplantıdır. Yapılan işlerin kalitesi, doğruluğu ve yanlışları değerlendirilir (Butt, 2022) . Şekil 3.2'de scrum yapısı gösterilmişir.



Şekil 3.2. Scrum yapısı.

3.3. Yazılım Test Süreci

Yazılım testi, sistemin belirli gereksinimlerin karşılandığını kontrol etmekle beraber aynı zamanda sistemin hedeflenen ortamında/ortamlarında kullanıcı ihtiyaçlarını karşılayıp karşılamadığını kontrol etmeyi amaçlamaktadır (ISTQB, 2018). Birimlerin ve sistemlerin uygulama sırasında hatalı çalışma riskini düşürmek amacıyla yazılım geliştirme yaşam döngüsü boyunca test faaliyetleri gerçekleştirilir. Yazılım yaşam döngüsü boyunca yapılan test faaliyetleri, hataların tespit edilmesini sağlar ve ardından test edilen bu birimlerin veya sistemlerin düzeltilmesini sağlayarak kaliteli yazılım ürünü ortaya çıkmasına katkıda bulunur. Uygun test aktivitelerinin gerçekleştirilebilmesi için kullanılan yazılım geliştirme yaşam döngüsü modelleri yapısını bilmek gerekmektedir.

Döngüsel yazılım geliştirme, ele alınan yazılım özelliklerinin genellikle belirli bir süreye sahip bir dizi döngüde belirlenmesi, tasarlanması, oluşturulması ve birlikte test edilmesi anlamına gelir (ISTQB, 2018).

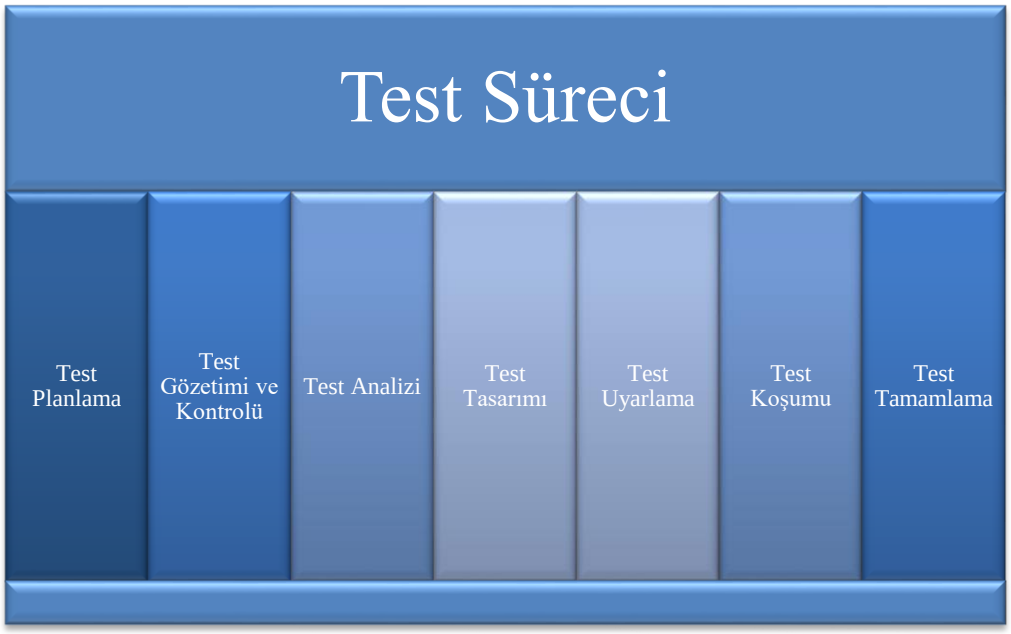
Döngüler, daha önceki döngülerde geliştirilen yazılım özelliklerindeki değişikliklerin yanı sıra proje kapsamındaki değişiklikleri de içerebilir. Son yazılım teslim edilinceye veya yazılım geliştirme durdurulana kadar her döngü, genel yazılım özellikleri kümesinin büyüyen bir alt kümesi olan çalışan yazılımı sunar. Hangi yazılım geliştirme yaşam döngüsü modeli seçilirse seçilsin, test faaliyetleri yaşam döngüsünün erken aşamalarında başlamalıdır.

Yazılım testinin genel hedefleri:

- Gereksinimler, kullanıcı senaryoları, yazılım/sistem tasarım ve kod gibi çalışma ürünlerini değerlendirmek
- Belirtilen tüm gereksinimlerin yerine getirilip getirilmediğini doğrulamak
- Test nesnesinin eksiksiz olup olmadığının ve kullanıcıların ve diğer paydaşların beklediği şekilde çalıştığını doğrulamak
- Test nesnesinin kalite seviyesi hakkında güven oluşturmak
- Hataları önlemek
- Arızaları ve hataları tespit etmek
- Paydaşlara test nesnesi ile ilgili sağlıklı veriler sunmak
- Yazılımın kalitesiz olma riskini düşürmek
- Sistem tasarım hatalarını azaltabilirmek için yapılacak testler ile erken aşamada tespit etmek
- Kod ve testlerde hata çıkma riskini azaltabilmek

3.3.1. Test süreci

Test süreci, proje kapsamı içerisinde belirtilen kriterlerin karşılanması için sıralı ve düzenli olarak yapılması gereken bir kılavuz özelliği taşımaktadır. İstenilen hedefe ulaşılması için test sürecinde belirtilen aşamalar bulunmaktadır. Test süreci aşamaları Şekil 3.3'de ifade edilmiştir (ISTQB, 2018).



Şekil 3.3. Test süreci.

Test Planlama: Test kapsamının belirlendiği aşamadır. Uygulanacak test teknikleri/seviyelerini ve test için gerekli yazılım/donanım araçları işbelirlenir. Test zaman çizelgesinin belirlendiği aşamadır.

Test Gözetimi ve Kontrolü: Test kontrolü, test planında belirlenen hedeflere ulaşmak için gerekli önlemlerin alınmasını içerir. Test gözetimi ve kontrolü, çıkış kriterlerinin değerlendirilmesiyle belirlenir.

Test Analizi: Test koşullarını tanımlandığı aşamadır. Test analizi ölçülebilir kapsam için hangi maddelerin test edileceği belirlenir.

Test Tasarımı: Test senayolarının hangi seviyede yazılması gerektiği belirlenir. Test verileri tanımlanır. Test senaryo seviye grupları ve test yazılımları gibi ayrıntılar belirlenir. Test senaryoları önceliklendiririlir ve test prosedürleri şablonu hazırlanır.

Test Uyarlama: Manuel testin gerçekleştirilmesi için test prosedürlerlerinin, test otomasyonu için test betiklerinin oluşturulduğu aşamadır. Test ortamının oluşturulması ve simülatör, otomasyon araçları gibi ihtiyaç duyulan her doküman ve aracın doğru şekilde kurulduğunun doğrulandığı adımdır.

Test Koşumu: Yazılan test senaryolarının manuel veya test betiklerininin otomasyon olarak yapıldığı adımdır. Test aşamasında gerçekleşen sonuçlar ile beklenen sonuçlar karşılaştırılır. Gözlemlenen hataların kayıt altına alındığı ve test sonucunun kaydedildiği adımdır.

Test Tamamlama: Proje boyunca yapılan tüm test prosedürlerinin ve elde edilen diğer ilgili bilgileri toplandığı ve kayıt altına alındığı aşamadır. Test sonucunda ortaya çıkan tüm hata sonuçlarının kapatılmadığının kontrol edildiği ve paydaşlara iletilmek üzere test kayıt formlarının ve raporlarının oluşturulduğu adımdır. Gelecek sürümler ve projeler için tamamlanan test faaliyetlerinden çıkarılan derslerin analiz edildiği aşamadır.

3.3.2. Test seviyeleri

Test seviyeleri, test aktivite gruplarıdır. Her test seviyesi, belirli bir yazılım geliştirme seviyesindeki yazılımla ilgili olarak gerçekleştirilen, bağımsız birimlerin, bileşenlerin veya sistemlerin testinden oluşur. Test seviyeleri, yazılım geliştirme yaşam döngüsü içindeki diğer aktivitelerle bağlantılıdır. Test seviyeleri temel olarak eksik alanları tanımlamak ve SDLC aşamaları arasındaki tekrarı önlemek içindir. Bu nedenle SDLC boyunca her aşamada kod doğrulamasını içn test uygulanmaktadır. Test seviyeleri aşağıda verilmiştir: (ISTQB, 2018)

• Birim testleri : Birbirinden bağımsız en küçük bileşenlerin veya modüllerin ayrı olarak test edilmesidir. Birim testi ile ortaya çıkan hatalar daha üst seviyelere ilerlemeden erken aşamada tespit edilmektedir. Özellikle kod değişikliklerinin sürekli olarak devam ettiği artımlı ve döngüsel geliştirme modellerinde otomatik birim regresyon testleri, yazılımdaki mevcut birimlerin kalitesine karşı güven oluşturur. Genellikle yazılım geliştirilirken yazılımcılar tarafından yapılmaktadır.

• Entegrasyon testleri: Yazılım geliştirildikçe ortaya çıkan birimlerin veya sistemlerin birbirine entegre edildiği zaman bir bütün olarak test edilmesidir. Birimler veya sistemler arasındaki etkileşime odaklanmaktadır. Arayüzlerin fonksiyonel ve fonksiyonel olmayan davranışlarının, tasarlandığı ve gereksinimde belirtildiği gibi olup olmadığının doğrulanması için gerçekleşmektedir.

• Sistem testleri: Bütün bir sistemin veya ürünün uçtan uca tüm gereksinimleri karşılayıp karşılamadığına ve uygulamadan beklenen şekilde çalışıp çalışmadığı test

edilir. Fonksiyonel ve fonksiyonel olmayan (performans, güvenlik, stres vb.) davranışlar ele alınır.

Kabul testler: Sistemin müşteriye (son kullanıcı) çıkmaya ve kullanıma hazır olduğunu değerlendirmeye yönelik bilgiler elde etmek için yapılır. Bütün bir sistemin

yeteneklerine odaklanılır. Kabul testi talep halinde müşteri ile birlikte de gereçekleşitirilebilmektedir.

3.3.3. Test çeşitleri

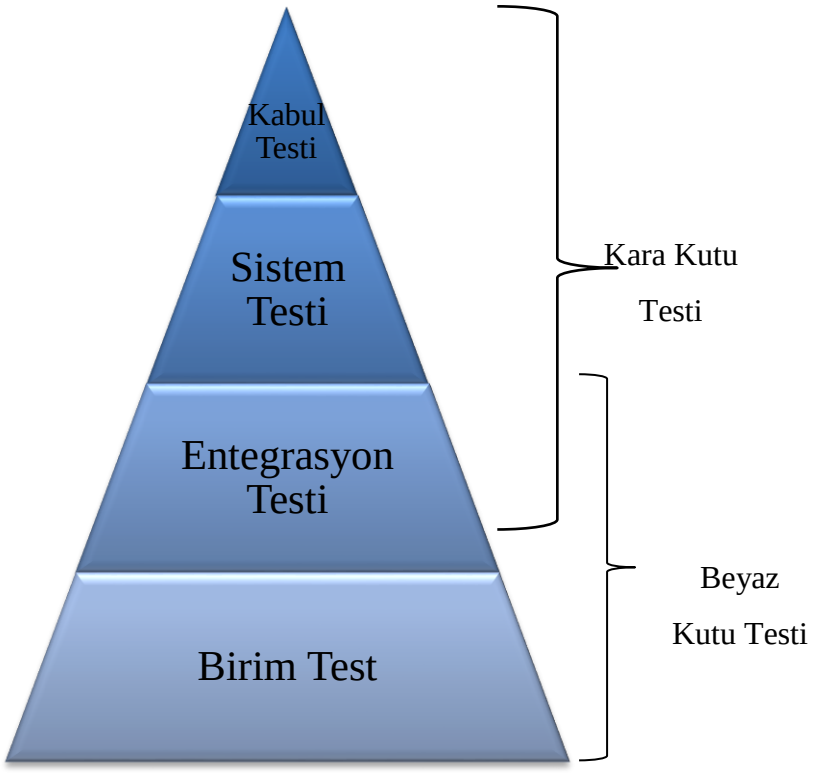
Test çeşidi, belirlenmiş test hedeflerine dayanarak bir yazılımın belirli özelliklerini veya bir sistemin bir bölümünü test etmeyi amaçlayan bir test aktiviteleri grubudur. Bu test çeşitleri aşağıdakileri içerebilir: (ISTQB, 2018)

Fonksiyonel testler: Bir sistemin fonksiyonel testleri, sistemin gerçekleştirmesi gereken fonksiyonları değerlendiren testleri içerir. Yazılım nasıl çalışması gerektiği ile ilgilenir. Fonksiyonel gereksinimler; iş gereksinimleri, kullanıcı gereksinimleri, betikler, kullanıcı hikâyeleri, kullanım senaryoları veya fonksiyonel spesifikasyonlar gibi çalışma ürünlerinde tanımlanmış olabilir. Fonksiyonlar gereksinimler sistemin ne yapması gerektiğini tanımlar.

Fonksiyonel olmayan testler: Bir sistemin fonksiyonel olmayan testleri, sistemlerin ve yazılımların, kullanılabilirlik, performans veya güvenlik gibi özelliklerini değerlendirir. Fonksiyonel olmayan testler sistemin yapılması gerekenleri "ne kadar iyi" yaptığını ölçümlemeye çalışır. Fonksiyonel olmayan testler genellikle otomasyon araçları ile test edilir.

Beyaz kutu testi: Yazılımın içyapısının (kodun) test edilmesidir. Bu yöntemdeki ana amaç kod parçacıklarının tek tek test edilerek aslında en küçük parçacık halinde bile sağlıklı bir şekilde çalıştırılabildiğinin görülmesidir. Geliştiriciler, yazılım gereksinimlerini karşılamak amacı ile tasarladıkları kod parçaları girdilerinin çıktıları karşılayıp karşılamadığı test eder. Yazılımın işlevselliği test edilmez.

Kara kutu testleri: Sistemin, yazılımın içyapısına (koda) bakılmadan sistemin işlevselliğini test edilmesidir. Yazılımın mimari yapı veya kaynak kod hakkında bilgi sahibi olmadan test uygulanabilir. Kara kutu testlerindeki amaç, gereksinimleri karşılayan çıktıların alınıp alınmadığını ölçmektir. Sistemden almayı beklediğimiz çıktılar kadar beklemediğimiz çıktılar da test edilmektedir. Şekil 3.4'de test seviyesi ve test çeşitleri ilişkisi gösterilmiştir.



Şekil 3.4. Test seviyesi.

Yazılım hataları ve yazılımın çalışma sistemine ait riskleri yönetilebilir kılmak, kodun ileriye dönük geliştirilme maliyetini azaltmak, ürün çalıştırılmadan önce uygunluğunu denetlemek, geliştirme sırasında gözden kaçan yanlışları bulmak ve bu yanlışların ileride de tekrarlanmasını önleyerek zaman ve maliyet tasarrufu yapılması için her aşamada yazılım testi uygulanmalıdır.

3.4. Bulanık SWARA

Değerlendirmeyi yapan karar vericilerin veya değerlendirilen kriterlerin ağırlıklarını belirlemek çok kriterli karar verme problemlerinin en önemli kısımlarındandır. Ağırlıklandırma için son yıllarda tercih edilen çok kriterli karar verme tekniklerinden biri de SWARA (Stepwise Weight Assessment Ratio Analysis) tekniğidir. SWARA yönteminin temel özelliği, kriterlerin ağırlıklarının belirlenme sürecinde, uzman görüşlerini öngörme yeteneğidir (Kersuliene ve ark. 2010). SWARA yöntemi, ağırlıklandırma için kullanılan diğer çok kriterli karar verme yöntemlerinden (AHP, BWM, vb) farklı bir bakış açısına sahiptir. Özellikle AHP yöntemine göre daha az ikili karşılaştırma gerektirmesinden dolayı daha tutarlı ve etkili kabul edilmektedir (Sheikh ve Mveal, 2023). SWARA yöntemi karar vericilerin belli bir ölçeğe bağlı kalmasını

beklemediğinden, karar vericilerin bakış açılarını daha esnek bir biçimde yansıtmalarına olanak sağlamaktadır. Ayrıca hesaplama karmaşıklığı da diğer yöntemlere göre daha azdır (Stanujkic ve ark. 2015; Agarwal ve ark. 2020).

SWARA yöntemi öznel bir teknik olmakla birlikte, uzmanın görüşlerini değerlendirme olanağı sağladığı için tercih edilmiştir. Aslında, öznelliğin etkisini azaltmak için SWARA tekniği bulanık sayılarla entegre edilerek kriter ağırlıkları hesaplanabilir. Bulanık küme teorisi (Zadeh, 1965), subjectif ve belirsiz problemlere yönelik bir yaklaşımdır ve genellikle verilerdeki esnekliğe ve belirsizliğe dayalı kararlar vermek için uygulanır (Pradhan ve ark, 2022). Bir bulanık küme hem dilsel hem de sayısal modelleme için çok yönlü bir araçtır. Bulanık kümenin avantajları nedeniyle, bu çalışmada yazılım test süreçlerinde meydana gelen hataların projeye yansıyabilecek farklı etkilerinin (etki alt bileşenleri ("zamana etki", "kaliteye etki", "maliyete etki", "çözülebilirliğe etki", "kullanılabilirliğe etki") ağırlıklarını hesaplamak için bulanık SWARA yöntemi kullanılmıştır. Bu çalışmada Yazdani ve Chang tarafından belirleniş bulanık sayılar kullanılarak Bulanık SWARA yöntemi adımları aşağıda açıklanmıştır: (Sengül ve Çağıl, 2020).

Adım1: Kriterler en önemliden başlamak üzere sıralanır.

Adım2: İkinci kriterden başlayarak, her bir kritere göreli önem düzeyi Yazdani ve Chang'ın kullanmış olduğu bulanık sayıların belirtildiği Tablo 3.2'e göre belirlenir (Chang, 1996). Bunun için, j kriteri ile bir önceki kriter (j-1) karşılaştırılır. Kersuliene ve arkadaşları tarafından bu orantıyı "ortalama değerin karşılaştırmalı önemi" diye adlandırılmış ve sj simgesi ile göstermiştir.

Tablo 3.2. Yazdani ve Chang'ın kullanmış olduğu bulanık üye fonksiyonu değerleri.

Değer	**Bulanık Sayı**	**Sıralama**
Çok Düşük	0,0,0.25	5
Düşük	0,0.25,0.5	4
Orta	0.25,0.5,0.75	3
Yüksek	0.5,0.75,1	2
Çok Yüksek	0.75,1,1	1

Adım 3: Katsayı (kj), denklem 3.1 ifadesi ile bulunur.

$$k_j = \begin{cases} \tilde{1} & j = 1 \\ \tilde{s_j} + \tilde{1} & j > 1 \end{cases} \quad (3.1)$$

Adım 4: Önem vektörü olan qj, değeri denklem 3.2 ile elde edilir:

$$\tilde{q}_j = \begin{cases} \tilde{1} & j = 1 \\ \frac{\tilde{x}_{j-1}}{\tilde{k}_j} & j > 1 \end{cases} \quad (3.2)$$

Burada, xj-1 notasyonu qj-1'e işaret etmektedir.

Adım 5: Kriterlerin ağırlık değerleri (wj), denklem 3.3 ile hesaplanır:

$$\widetilde{w}_j = \frac{\tilde{q}_j}{\sum_{k=1}^{n} \tilde{q}_k} \quad (3.3)$$

$\widetilde{w}_j$, j. inci kriterin bulanık ifade ile önemini göstermektedir. Hesaplamalar süresince ifadeler üçgensel bulanık sayılar ile l1 ≤ m1 ≤ u1 olacak şekilde A1= (l1, m1, u1) şeklinde gösterilecektir.

Adım 6: Kriterler ağırlıkları (wj) bulanık yapıda olduğundan durulaştırma işlemi Kiani ve arkadaşlarınında (2017) çalışmalarında bahsetmiş olduğu eşitlik denklem 3.4 ile elde edilir:

$$w_j = \frac{\left(w_j^u - w_j^l\right) + \left(w_j^m - w_j^l\right)}{3} + w_j^l \quad (3.4)$$

Moslem ve arkadaşları (2023) çalışmalarında şehir içi otobüs taşımacılığında tedarik kalitesini değerlendirmek için belirledikleri üç ana ve 21 alt kriteri ağırlıklandırmak için SWARA yöntemini kullanmışlardır. Saeidi ve arkadaşları (2022) çalışmalarında sürdürülebilir insan kaynakları yönetiminde kullanılacak kriterleri önceliklendirmek için SWARA yöntemini kullanmışlardır. Yücenur ve Şenol (2021) inşaat sektöründe atıkların ortadan kaldırılması için SWARA yöntemi uygulamışlardır.

3.5. Hata Türü ve Etkileri Analizi

Hata Türü ve Etkileri Analizi (FMEA), bir üründe veya bir sisteminde gerçekleşen hata tespiti ve bu durumun yarattığı hata etkisinin azaltılması için kullanılan sistematik bir yaklaşım olan risk değerlendirme yöntem aracıdır (Yener ve Can, 2021). FMEA, 1949 yılında Amerika Birleşik Devletleri Askeri prosedürü MIL-P-1629 ile ortaya çıktımıştır (Pentti ve Atte, 2002). FMEA'nın temel amacı, bir ürünün/sürecin potansiyel hatalarını tanımlamak ve bunun etkilerini azaltabilecek güvenli düzenleyici eylemleri uygulamak için riskleri tahmin etmek ve değerlendirmektir. Günümüzde FMEA , havacılık, imalat, otomotiv kimya, denizcilik, tıp, nükleer gibi birçok alanda uygulanmaktadır (Liu ve ark, 2015).

Klasik FMEA'da, Hata modlarının tanımlanması ve sıralanması, üç belirleyici risk bileşeni "etki (E), olasılık (O) ve farkedilebilirlik (F)" çarpımı ile bulunan RÖS değeri ile belirlenir (Rezaee ve ark, 2020).

$$RÖS=0\times E\times F \tag{3.5}$$

Risk olasılık, etki ve farkedilebilirlik skalası belirlenen 10 puanlık ölçek skalasına göre oluşturulmuştur (Erbay, 2019). Risk olasılık değer skalası Tablo 3.3 'de, risk etki değer skalası Tablo 3.4'de ve farkedilebilirlik değer skalası Tablo 3.5'de belirtilmiştir.

Tablo 3.3. Olasılık değerinin skalası.

Olasılık Tanımı	Olasılık	Değer
	> 1/2	10
Çok Yüksek	1/3	9
	1/8	8
Yüksek	1/20	7
	1/80	6
Orta	1/400	5
	1/2000	4
Düşük	1/15000	3
	1/150000	2
Çok Düşük	<1/150000	1

Tablo 3.4. Etki değerinin skalası.

Etki	Değer
Çok Tehlikeli	10
Tehlikeli	9
Çok Yüksek	8
Yüksek	7
Orta	6
Düşük	5
Çok Düşük	4
Küçük	3
Çok Küçük	2
Yok	1

Tablo 3.5. Farkedilebilirlik değerinin skalası.

Farkedilebilirlik	Değer
İmkansız	10
Çok zor	9
Zor	8
Çok az	7
Az	6
Orta	5
Ortanın üstü	4
Yüksek	3
Çok yüksek	2
Hemen	1

Bir çok araştırmaya göre geleneksel FMEA'nın risk değerlendirme sürecinde bazı zayıf yönleri vardır. Liu ve arkadaşlarına göre (2015), FMEA'nın eksiklikleri şu şekildedir:

1. Üç belirleyici bileşen (etki (E), olasılık (O) ve farkedilebilirlik (F))'in farklı kombinasyonları aynı RP değeri verdiğinde aynı öneme sahip risk değerleri ortaya çıkar ve durum önceliklendirme zorlaştıracaktır.
2. Risklerin bir sistemde farklı derecede E,O,F değerleri vardır. Bu nedenle farklı riskler için aynı E,O,F ağırlığı vermek uygun değildir.
3. Üç risk bileşen aynı öneme sahip olduğu varsayılır, gerçek uygulamalarda göreceli önem vardır.
4. Uzmanlar tarafından üç risk faktörü tamsayı değerleri verilerek hesaplanır.

Bu araştırma sonuçlarına bakıldığında geleneksel FMEA'nın tek başına kullanımı risk değerlendirmesi için çok sağlıklı sonuçlar ortaya çıkarmayacağı görülmektedir. Bu nedenle çok kriterli karar verme yöntemleri geleneksel FMEA'nın eksiklerini iyileştirmek için kullanılmaktadır. Klasik FMEA'da yer alan nitel ifadelerin nicel bilgilere dönüşmesinde ÇKKV yöntemlerinin bulanıklığından yararlanılmışdır. Bu noktada ÇKKV, risklerin ağırlıklandırma, önceliklendirme ve sıralamada geleneksel FMEA'nın optimize edilmesinde rol oynamıştır.

Bu çalışmada, FMEA'nın üç bileşenden biri olan etkinin ağırlıklandırılması için ÇKKV tekniklerinden Bulanık SWARA yöntemi kullanılmış ve özgün "Ağırlıklandırılmış FMEA" modeli önerilmiştir.

3.6. Ağırlıklandırılmış Hata Modu Etki Analizi

Klasik FMEA uygulamaları, uzmanların risk bileşenlerinin farklı hata modlarının üzerindeki önemini tam olarak anlamasını zorlaştırmaktadır. Bu zorluğu ortadan kaldırmak için risk bileşenlerinin ağırlıkları çok kriterli karar verme teknikleri kullanılarak hesaplanabilir. Uzmanların sezgi ve deneyimlerini değerlendirmeye katabilmek için ise son yıllarda FMEA uygulamaları belirsiz bilgileri işleyebilen Bulanık Küme teorisi (Zadeh, 1695) ile birlikte kullanılmaya başlanmıştır. Liu ve arkadaşları (2015) FMEA'nın dezavantajlarını azaltmak için Intuitionistic Fuzzy Hybrid TOPSIS yaklaşımı ile entegre şekilde kullanmıştır. Kahraman ve arkadaşları

(2013) sağlık hizmetlerindeki hataları önceliklendirmek için FMEA tekniğini Kural Tabanlı Bulanık Mantık yaklaşımı ile birlikte kullanmışlardır.

Klasik FMEA'nın literatürde bahsedilen eksikliklerinin yanı sıra, risk değerlendirme yöntemi olarak riskleri önceliklendirirken kullanılan risk bileşenlerinden "etki" bileşeninin tam olarak neyi önceliklendirdiği çok açık değildir. Kaliteye etkiyi mi? Maliyete etkiyi mi? Teslim süresine etkiyi mi? gibi. Eğer risk sadece bir alanı etkiliyorsa sıkıntı olarak görülmeyecek bu durum, risk farklı alanları farklı ağırlıklarla etkilediğinde problem oluşturabilecektir. Bu nedenle bu çalışmada klasik FMEA'nın risk bileşenlerinden "etki" bileşeni örnek uygulamanın yapıldığı yazılım sektörü de

düşünülerek beş alt bileşene ayrılmıştır ("zamana etki", "kaliteye etki", "maliyete etki", "çözülebilirliğe etki", "kullanılabilirliğe etki" şeklinde). Bu beş alt risk bileşeni ile birlikte risk önceliğini hesaplamak için kullanılan RÖS'ün geliştirilmiş yeni hali denklem 3.6' de gösterildiği gibi olmuştur.

$$\text{ARÖS}=O\times E(E_1\times E_2\times E_3\times E_4\times E_5)\times F \tag{3.6}$$

(ARÖS: Ağırlıklveırılmış risk öncelik numarası; O: Hatanın Olasılık Değeri ; E_1: Zaman Bileşinin Etki Değeri; E_2: Kalite Bileşinin Etki Değeri; E_3: Maliyet Bileşinin Etki Değeri; E_4: Çözülebilirlik Bileşinin Etki Değeri; E_5: Kullanılabilirlik Bileşinin Etki Değeri ; F: Hatanın Farkedilebilirlik Değeri)

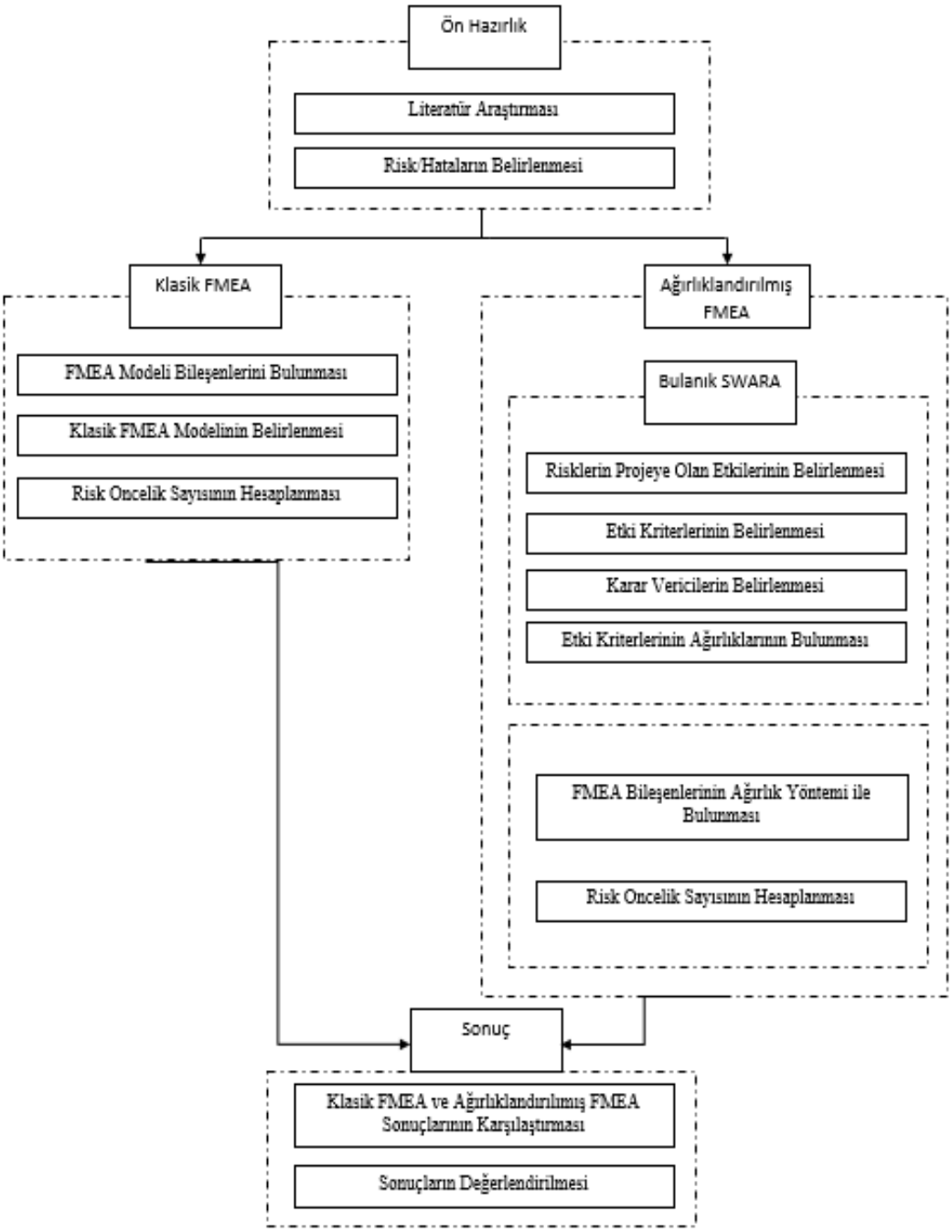
Önerilen risk değerlendirme modelinin sonraki aşamasında, risk öncelik sayısını veren formüldeki her bir risk bileşeni ve alt bileşeni (FMEA yönteminin her bir bileşenin aynı ağırlığa sahip olması dezavantajını ortadan kaldıracak şekilde) Bulanık SWARA yöntemi ile ağırlık verilmiştir.

4. UYGULAMA

Çalışmanın bu bölümünde Ankara ilinde bulunan yazılım alanında faaliyet gösteren bir savunma sanayi şirketinde gerçekleşen haberleşme projesi için çok kriterli karar verme yöntemlerinden bulanık Adım Adım Ağırlık Değerlendirme Oran Analizi (SWARA) ile Hata Türü ve Etkileri Analizi yönteminden yararlanılarak yeni bir risk analizi yöntemi uygulanmıştır. Haberleşme projesi boyunca yazılım test süreçlerinde ortaya çıkan riskler için uygulama gerçekleştirilmiştir.

Yazılım testi, geliştirilen yazılımın belirtilen gereksinimleri istenilen şekilde karşılayıp karşılamadığını bulmak amacıyla bir yazılım uygulamasının fonksiyonetiliğini değerlendirmek ve ürünü hatasız ya da en az hatayla ortaya çıkarmak için yapılan sürecin adıdır. Yazılım testi, sadece mevcut yazılımdaki hataları bulmayı değil, aynı zamanda yazılımı kaliteli, verimli, doğru bir şekilde kullanmayı ve kullanılabilirlik açısından geliştirecek özellikler bulmayı amaçlar.

Yazılım testi, manuel olarak veya otomasyon araçları ile yapılmaktadır. Testin otomasyon veya manuel olarak yapılmasına yazılım ürünün özellikleri ve doğrulama kriterlerine göre karar verilmektedir. Bu çalışmada manuel yapılan testler ele alınmıştır. Manuel test, otomasyona göre daha fazla insan gücü gerektiren bir süreç olması nedeni ile zaman açısından uzun sürebilmektedir. Yazılım testinin yapılması genellikle zaman kaybı gibi gözükse de hatayı erken tespit etme ve hatanın önlenmesini sağladığı için uzun vadede maliyeti düşürmektedir. Proje boyunca yapılan her testin sonucunda ortaya çıkan hatalar, iyileştirmeler yazılım sektöründe çok kullanılan Redmine, Jira gibi araçlar yardımı ile kayıt altına alınır. Bu araçlar üzerinden aynı hatanın kaç kere tekrar ettiği, hangi sürümlerde meydana geldiği, testin gerçekleşme süresi, hatanının kapanma tarihi veya gerçekleşen hatanın başka bir hataya neden olup olmadığı gibi, güncel durum takip edilir. Bu çalışmada uygulanacak yöntemlerin akışı Şekil 4.1'de verilmiştir.



Şekil 4.1. Çalışma akış şeması.

4.1. Hataların Belirlenmesi

Her sprint sonu gerçekleştirilen tesler ile ortaya çıkan hatalar Redmin/Jira gibi dijital araçlarda kayıt altına alınmaktadır. Bu hataların hangisinin çok tekrar ettiği "Reopen" durumlarına göre kontrol edilmektedir. Bu uygulamda test kayıtlarında en çok tekrar eden hataların bir kısmı ele alınmıştır. Ele alınan veriler aşağıda listelenmiştir.

Tablo 4.1. Hata listesi.

No	Hata
H1	Uygulamanın açılmaması
H2	Uygulama kullanılırken bir süre sonra yavaşlaması
H3	Uygulamaya ilk girişde yetkisiz kullanım uyarı mesajının alınması
H4	Uygulamanın donması
H5	Lokalde uygulamanın çalışması ancak kullanıcı bilgisayarında erişimin sağlanamaması
H6	URL adresinde yanlış IP bilgisinin yer alması
H7	Uygulamanın kamera erişimine izin vermemesi
H8	Kullanıcılar arası görüşmelerde ses iletiminin gerçekleşmemesi
H9	Çok fazla işlem peş peşe yapıldığında uygulamanın kullanılmaz hale gelmesi
H10	Görüntülü ve sesli görüşmelerde bağlantının gidip gelmesi
H11	Uygulamaya giriş parola geçerlilik süresi bitmemesine rağmen uyarı mesajının gösterilmesi
H12	Uygulamaya entegre edilmiş diğer uygulamara geçişin sağlanamaması
H13	Profil fotoğraf önizlemesinin gelmemesi
H14	Kullanıcının pasif olarak kaydedilememesi
H15	Aktif olan kullanıcının yasaklanmasına rağmen silinmiş gibi gözükmesi
H16	İlk kullanıcı oluşturma işleminin başarısız ikinci oluşturma işleminin başarılı olması

Tablo 4.1. (Devam): Hata listesi.

No	Hata
H17	Başarısız işlem uyarı mesajının alınması
H18	İşlem kayıtları alanında dil özelliklerinin doğru çalışamaması
H19	Yeni bir kayıt yapılırken ekranın eski veriler ile birlikte açılması
H20	Gruplara birden fazla kullanıcının atanamaması
H21	Anahtar geçerlilik süresi alanına 0 değerinin girilmesi
H22	Ayarlar ekranındaki varsayılan değerlerin doğru gösterilmemesi
H23	Aynı kullanıcının birden fazla oturum açması
H24	Parola alanı maskeleme özelliğinin çalışmaması
H25	Kullanıcı profil alanında kullanıcı bilgilerinin gösterilmemesi
H26	Aynı eposta adresine sahip birden fazla kullanıcının kaydedilmesi
H27	Log kayıtlarının oluşturulamaması
H28	Güncelenen şifrenin kullanılamaması
H29	Uygulama açılış sayfasının yüklenememesi
H30	Anahtar geçerlilik süresi alanınan "-" değerlerin girilmesi
H31	Kullanıcı oluştururken girilen anahtar başlangıç tarihinin kayıt yapıldıkdan sonra yarın başlayacak şekilde göstermesi
H32	Uygulmadaki kullanıcı görüntü iletiminde kopmaların olması
H33	Belge yükleme butonu yazılarınının okunamaması
H34	Port bilgisinin sürekli değişmesi
H35	Kullanıcıya yetki verildiğinde uygulamanın kullanılmaz hale gelmesi
H36	İşlem kayıtları alanında listedeki herhangi bir alana basıldığında tüm kayıtların listelenmesi

Tablo 4.1. (Devam): Hata listesi.

No	Hata
H37	Kullanıcı mesajlaşma alanında gönderilen mesajlaringelen mesajlar alanında gösterilmesi
H38	Sesli ve görsel uyarı mesajlarının alınamaması
H39	Kıdem adı alanına girilen verinin, sayısal değer olarak gösterilmesi
H40	Silinen anahtar tanımı ile aynı tanımda yeni anahtarın oluşturulaması
H41	Kullanıcının oluşturulamaması
H42	Kullanıcı anahtar bağı koparma işleminin yapılamaması
H43	Uyarı mesajının yanlış gösterilmesi
H44	Hesap kilitleme süresi sonrasında doğru şifre ile uygulamaya giriş yapılamaması
H45	Uygulamada işlem yapılırken uygulamanın, giriş ekranına atması
H46	Kullanıcılara atanan görevlerin refreshlemeden gösterilmemesi
H47	Logo kullanım hatası
H48	Uygulamaya giriş ekranında versiyon bilgisinin gösterilmemesi

4.2. Klasik FMEA Yöntemi ile Risk Öncelik Değerinin Bulunması

Test kayıtlarında yer alan hataların önceliklendirme işlemi için ilk olarak klasik FMEA yöntemi kullanılmışdır. Tablo 4.2'de hataların olasılık (O), etki (E) ve farkedilebilirik (F) değerleri belirlenmiş ve bu değerlere göre RÖS değeri hesaplanmıştır.

Tablo 4.2. Klasik FMEA.

No	Olasılık	Farkedilebilirlik	Etki	RÖS
H1	5	9	9	405
H2	7	6	7	294
H3	3	9	5	135
H4	5	8	8	320

Tablo 4.2. (Devamı): Klasik FMEA.

No	Olasılık	Farkedilebilirlik	Etki	RÖS
H5	8	6	6	288
H6	3	5	6	90
H7	7	7	7	343
H8	7	7	7	343
H9	5	9	8	360
H10	8	6	7	336
H11	3	4	6	72
H12	7	6	7	294
H13	9	4	5	180
H14	7	8	6	336
H15	7	8	6	336
H16	5	7	7	245
H17	3	8	7	168
H18	7	5	4	140
H19	6	5	6	180
H20	2	3	4	24
H21	9	3	5	135
H22	8	3	5	120
H23	6	5	7	210
H24	6	5	2	60
H25	8	6	5	240
H26	7	5	7	245
H27	6	6	8	288
H28	4	6	8	192
H29	5	4	7	140
H30	8	6	5	240
H31	5	7	7	245
H32	5	8	7	280
H33	6	3	3	54
H34	3	8	9	216

Tablo 4.2. (Devamı): Klasik FMEA.

No	Olasılık	Farkedilebilirlik	Etki	RÖS
H35	6	8	7	336
H36	4	8	7	224
H37	5	7	6	210
H38	7	5	5	175
H39	3	8	7	168
H40	4	6	6	144
H35	6	8	7	336
H36	4	8	7	224
H37	5	7	6	210
H38	7	5	5	175
H39	3	8	7	168
H40	4	6	6	144
H41	4	8	8	256
H42	3	5	5	75
H43	8	7	6	336
H44	6	6	8	288
H45	8	5	8	320
H46	8	6	6	288
H47	2	3	4	24
H48	7	4	5	140

4.3. Etki Alt Bileşenlerinin Ağırlık Değerinin Bulunması

Yapılan çalışmalarda klasik FMEA'nın eksikleri göz önünde bulundurulduğunda ve her hatanın projeye olan etkileri birbirinden farklı olacağı öngörüldü ve hataların projeye olan etkileri beş alt bileşende tanımlanmıştır. Beş etki bileşeni aşağıda belirtilmiştir.

- Zaman
- Kalite
- Maliyet (AdamxAy)
- Çözülebilirlik
- Kullanılabilirlik

Belirlenen bu bileşenlerin ağırlıklarının belirlenmesi için karar verici görüşlerine daha çok önem veren ÇKKV tekniklerinden olan Bulanık SWARA yönteminden yararlanılmıştır. Bu çalışmada teknik yönetici, yazılım/yazılım test mühendisi, ve kalite uzmanı olmak üzere üç uzmanın görüşü alınarak uygulama gerçekleştirilmiştir. Tablo 4.3'de karar vericilerin en önemli olan bileşenden başlayarak yapmış oldukları sıralama gösterilmektedir.

Tablo 4.3. Karar vericilerin önem sıralaması.

Bileşen	**KV1-(Yönetici)**	**KV2-(Yazılımcı)**	**KV3-(Kalite)**
Zaman	1	1	1
Kalite	3	4	2
Maliyet(Adam*Ay)	2	5	4
Çözülebilirlik	5	2	5
Kullanılabilirlik	4	3	3

Tablo 4.4'de, karar vericilerin belirlediklerin sıralamaya göre bileşenlerin bulanık üçgen sayı matrisi ve pj ortalama değerleri gösterilmektedir.

Tablo 4.4. Bulanık üçgen sayı matrisi.

Üçgen Sayı	**Zaman**	**Kalite**	**Maliyet**	**Çözülebilirlik**	**Kullanılabilirlik**
kv1l	0,75	0,25	0,5	0	0
kv1m	1	0,5	0,75	0	0,25
kv1u	1	0,75	1	0,25	0,5

Tablo 4.4. (Devamı) Bulanık üçgen sayı matrisi.

Üçgen Sayı	Zaman	Kalite	Maliyet	Çözülebilirlik	Kullanılabilirlik
kv2l	0,75	0	0	0,5	0,25
kv2m	1	0,25	0	0,75	0,5
kv2u	1	0,5	0,25	1	0,75
kv3l	0,75	0,5	0	0	0,25
kv3m	1	0,75	0,25	0	0,5
kv3u	1	1	0,5	0,25	0,75
pjl	0,75	0,25	0,166667	0,166666667	0,166666667
pjm	1	0,5	0,333333	0,25	0,416666667
pju	1	0,75	0,583333	0,5	0,666666667

Tablo 4.5'de ortalama değerin karşılaştırmalı önemi (sj), katsayı değer (kj), önem vektörü (qj) hesaplanmıştır.

Tablo 4.5. Katsayı ve önem vektörü.

	Ortalama değerin karşılaştırmalı önemi (Sj)			Katsayı (Kj)			Önem Vektörü (qj)		
	sjl	**sjm**	**sju**	***kjl***	***kjm***	***kju***	***qjl***	***qjm***	***qju***
Zaman	0	0	0	1	1	1	1	1	1
Kalite	0,5	0,5	0,25	1,5	1,5	1,25	0,6667	0,6667	0,8
Maliyet(Adam*Ay)	0,0833	0,1667	0,1667	1,0833	1,1667	1,1667	0,6154	0,5712	0,6855
Çözülebilirlik	0	0,083	0,083	1	1,083	1,083	1	0,5274	0,6329
Kullanılabilirlik	0	-0,166	-0,166	1	0,833	0,833	1	0,6329	0,7595

Tablo 4.6’ da bileşenlerin ağırlık değeri denklem 3.3’e göre hesaplanmıştır.

Tablo 4.6. Bileşenlerin ağırlık değeri.

Bileşenlerin Ağırlıkları (wj)			
Bileşen	**wjl**	**wjm**	**wju**
Zaman	0,233532	0,294243	0,257848
Kalite	0,155688	0,196162	0,206278
Maliyet(Adam*Ay)	0,143716	0,168134	0,176805
Çözülebilirlik	0,233532	0,155206	0,16321
Kullanılabilirlik	0,233532	0,186254	0,195859

Tablo 4.7’ de, denklem 3.4’e göre etki bileşenlerinin normalleştirilmiş ağırlığı gösterilmektedir.

Tablo 4.7. Bileşenlerin normalleştirilmiş ağırlığı.

Bileşen	**Durulaştırılmış Ağırlık**
Zaman	0,26
Kalite	0,19
Maliyet(Adam*Ay)	0,16
Çözülebilirlik	0,18
Kullanılabilirlik	0,21

Tablo 4.7’de uygulanan Bulanık SWARA yöntemine göre Zaman bileşeninin ağırlığı 0.26, Kalite bileşeninin ağırlığı 0.19, Maliyet bileşeninin ağırlığı 0.16, Çözülebilirlik

bileşeninin ağırlığı 0.18, Kullanılabilirlilik bileşeninin ağırlığı 0.21 olarak hesaplanmıştır.

4.4. Ağırlıklandırılmış FMEA

Tezde ağırlıklandırılmış FMEA yöntemi uygulanmışdır. Bu aşamada olasılık ve farkedilebilirlik bileşenlerinin ağırlıkları eşit (1) olarak kabul edilmiş ve etki alt bileşenleri ağırlıklandırılmıştır. Riskin etki değeri ise bulanık SWARA ile bulunan etki alt bileşenleri ağırlıkları çarpımı ile bulunmuştur. Yazılım test süreçlerindeki risklerin risk öncelik numarasını belirlemek için denklem 3.6 kullanılmıştır.

Ağırlıklandırılmış risk öncelik numarası; O: Hatanın Olasılık Değeri; E_1: Zaman Bileşinin Etki Değeri; E_2: Kalite Bileşinin Etki Değeri; E_3: Maliyet Bileşinin Etki Değeri; E_4: Çözülebilirlik Bileşinin Etki Değeri; E_5: Kullanılabilirlik Bileşinin Etki Değeri; F:Hatanın Farkedilebilirlik Değeri bileşenleri ile hesaplanmıştır.

Tablo 4.8' de ağırlıklandırılmış FMEA yöntemi ile RÖS değerleri hesaplanmışdır.

Tablo 4.8. Ağırlıklandırılmış FMEA.

Hata Türü	Olasılık	Farkedilebilirlik	Etki					RÖS
			Zaman	Kalite	Maliyet	Çözülebilirlik	Kullanılabilirlik	
Ağırlık	1	1	0.26	0.19	0.16	0.18	0.21	
H1	5	9	9	7	9	8	8	487,88
H2	7	6	7	7	7	6	6	154,95
H3	3	9	5	7	5	8	7	79,05
H4	5	8	8	7	6	6	7	168,65
H5	8	6	8	6	5	7	7	168,67
H6	3	5	3	5	2	6	6	4,84
H7	7	7	5	6	4	6	7	73,78
H8	7	7	8	7	8	6	7	275,46
H9	5	9	9	7	8	7	8	379,46

Tablo 4.8. (Devamı) Ağırlıklandırılmış FMEA.

Hata Türü	Olasılık	Farkedilebilirlik	Zaman	Kalite	Maliyet	Çözülebilirlik	Kullanılabilirlik	RÖS
H10	8	6	7	8	5	8	8	256,99
H11	3	4	5	7	4	6	7	21,08
H12	7	6	6	7	5	6	7	110,68
H13	9	4	2	5	2	5	5	5,38
H14	7	8	6	6	4	5	6	72,28
H15	7	8	6	7	5	6	7	147,57
H16	5	7	5	7	5	6	8	87,84
H17	3	8	1	5	1	4	3	0,43
H18	7	5	6	8	6	6	8	144,56
H19	6	5	5	7	5	6	8	75,29
H20	2	3	6	7	4	7	8	16,87
H21	9	3	1	3	1	3	3	0,22
H22	8	3	1	3	1	2	2	0,09
H23	6	5	5	7	6	8	7	105,41
H24	6	5	1	3	1	3	2	0,16
H25	8	6	6	7	4	6	7	101,19
H26	7	5	7	7	7	7	8	200,86
H27	6	6	6	7	6	7	6	113,84
H28	4	6	6	6	6	6	7	65,05
H29	5	4	8	7	7	7	9	147,57
H30	8	6	1	4	1	3	4	0,69
H31	5	7	8	8	6	6	8	192,74

Tablo 4.8. (Devamı) Ağırlıklandırılmış FMEA.

Hata Türü	Olasılık	Farkedilebilirlik	Zaman	Kalite	Maliyet	Çözülebilirlik	Kullanılabilirlik	RÖS
H32	5	8	8	7	6	8	7	224,86
H33	6	3	2	6	3	3	5	2,9040
H34	3	8	8	8	6	7	8	154,19
H35	6	8	8	8	7	7	8	359,78
H36	4	8	7	7	7	7	6	137,73
H37	5	7	4	8	4	5	6	40,154
H38	7	5	3	6	4	4	5	15,058
H39	3	8	2	5	3	4	6	5,1627
H40	4	6	6	6	5	5	6	38,720
H41	4	8	8	8	7	8	9	308,38
H42	3	5	6	7	4	6	7	31,621
H43	8	7	1	6	2	2	5	2,0077
H44	6	6	7	8	6	6	9	195,15
H45	8	5	8	8	7	7	9	337,30
H46	8	6	5	6	4	5	6	51,627
H47	2	3	1	7	5	5	5	1,5685
H48	7	4	1	3	1	2	1	0,0501

5. SONUÇ VE ÖNERİLER

Teknolojinin gelişmesiyle birçok sektörde işlerin yönetilmesi, kontrol edilmesi ve faaliyetlerin geliştirilmesi için yazılım ve yazılım ürünlerine ihtiyaç gün geçtikçe artmıştır. Şirketlerin veya müşterilerin ihtiyaçlarını karşılayacak uygun yazımlıların / uygulamaların ortaya çıkması için uygulamanın son kullanıcıya ulaşmadan önce yazılım geliştirme aşamasında ayrıntılı olarak test edilmesi gerekmektedir. Bu nokta yazılım test süreçleri önem kazanmaktadır. Bir uygulamanın / yazılımın kalitesi onun geliştirme aşamasında sürekli olarak testten geçmesiyle ortaya çıkmaktadır. Ancak test sürecinde ortaya çıkan hataların çok fazla tekrar etmesi, her hatanın projeye olan etkisinin farklı olması nedeniyle çözüm aşamasında önceliklendirme yapılması gerekmektedir.

Bu tezde Ankara'da yer alan bir savunma firmasının yazılım projesindeki yazılım test süreçlerde ortaya çıkan risklerin çok kriterli karar verme tekniklerinden Bulanık SWARA'dan yararlanılarak yeni bir FMEA çalışması yapılmıştır. Yazılım test sürecindeki hatalar belirlendikten sonra ilk olarak klasik FMEA yöntemi uygulanmıştır. Bunun sonucunda her hatanı projeye olan etkileri farklı olacağı göz önünde bulundurulmuş ve kalsik FMEA'daki "Etki" bileşini, beş alt bileşene ayrılmıştır. Belirlenen beş alt bileşenin proje üzerindeki etkilerinin belirlenmesi için uzman görüşlerinden yararlanılarak Bulanık SWARA yöntemi kullanılmış ve beş alt etki bileşeninin ağırlık değerleri belirlenmiştir. Bu beş alt bileşenin etki değerleri ile ağırlıklarının çarpımı ile hesaplanan FMEA'ya ise Ağırlıklandırılmış FMEA yöntemi olarak ifade edilmiştir. Yapılan her iki çalışmada ilk onda yer alan hata türlerinin risk öncelik sayısı değerleri Tablo 5.1'de gösterilmektedir.

Tablo 5.1. Klasik FMEA ve ağırlıklandırılımış fmea sonuçlarının karşılaştırması.

Sıra No	Hata No	Hata Türü	Klasik FMEA RPN	Hata No	Hata Türü	Ağırıklandırımış FMEA RÖS
1	H1	Uygulamanın açılmaması	405	H1	Uygulamanın açılmaması	487,88
2	H9	Çok fazla işlem peş peşe yapıldığında uygulamanın kullanılmaz hale gelmesi	360	H9	Çok fazla işlem peş peşe yapıldığında uygulamanın kullanılmaz hale gelmesi	379,46
3	H7	Uygulamanın kamera erişimine izin vermemesi	343	H35	Kullanıcıya yetki verildiğinde uygulamanın kullanılmaz hale gelmesi	359,79
4	H8	Kullanıcılar arası görüşmelerde ses iletiminin gerçekleşmemesi	343	H45	Uygulamada işlem yapılırken uygulamanın, giriş ekranına atması	337,30
5	H10	Görüntülü ve sesli görüşmelerde bağlantının gidip gelmesi	336	H41	Kullanıcının oluşturulamaması	308,39
6	H14	Kullanıcının pasif olarak kaydedilememesi	336	H8	Kullanıcılar arası görüşmelerde ses iletiminin gerçekleşmemesi	275,46
7	H15	Aktif olan kullanıcının yasaklanmasına rağmen silinmiş gibi gözükmesi	336	H10	Görüntülü ve sesli görüşmelerde bağlantının gidip gelmesi	256,99

Tablo 5.1. (Devamı) Klasik FMEA ve ağırlıklandırılımış fmea sonuçlarının karşılaştırması.

Sıra No	Hata No	Hata Türü	Klasik FMEA RPN	Hata No	Hata Türü	Ağırıklandırımış FMEA RÖS
8	H35	Kullanıcıya yetki verildiğinde uygulamanın kullanılmaz hale gelmesi	336	H32	Uygulmadaki kullanıcı görüntü iletiminde kopmaların olması	224,87
9	H43	Uyarı mesajının yanlış gösterilmesi	336	H26	Aynı eposta adresine sahip birden fazla kullanıcının kaydedilmesi	200,86
10	H4	Uygulamanın donması	320	H44	Hesap kilitleme süresi sonrasında doğru şifre ile uygulamaya giriş yapılamaması	195,15

Söz konusu olan savunma şirketinde geliştirile uygulamaların yazılım testindeki hataların önceliklendirilmesi için yapılan ağırlıklveırılmış FMEA hesaplamasına göre risk önceliklendirme numarası incelendiğinde; uygulamanın açılmaması, çok fazla işlem peş peşe yapıldığında uygulamanın kullanılmaz hale gelmesi, kullanıcıya yetki verildiğinde uygulamanın kullanılmaz hale gelmesi en büyük değere sahip oldukları belirtilmiştir. Tablo 5.2'de, ağırlıklandırılmış FMEA değerlendirmesine göre yazılım test süreçlerinde ortaya çıkan hatalar için olası hafifletme/çözüm önerileri sunulmuştur.

Öneri olarak; hataların önceliklendirilmesi için karar verici mekanizmalar üzerinde yapılacak olan bir hibrit bir çalışma ile bu süreç için bir yazılım geliştirilebilir.

Tablo 5.2. Hata modu etki analizi.

No	Bileşen	Hata	Neden	Olasılık	Farkedilebilirlik	Zaman	Kalite	Maliyet	Çözülebilirlik	Kullanılabilirlik	RÖS	Hafifletme Önerileri
H1	Uygulama Yazılımı	Uygulamanın açılmaması	Sunucunun çalışmaması	5	9	9	7	9	8	8	487,8814	Test ortamının kontrol edilmesi. Bakım
H2		Uygulama kullanılırken bir süre sonra yavaşlaması	Sununcunun hizmet vermemesi	7	6	7	7	7	6	6	154,9475	Bakım
H3		Uygulamaya ilk girişde yetkisiz kullanım uyarı mesajının alınması	Güvelik duvarı yanlış yapılveırılması	3	9	5	7	5	8	7	79,05486	Konfigürasyon işlemleri prosedür halinde oluşturulmalı
H4		Uygulamanın donması	İşletim sisteminin çok yavaş çalışması ve daha sonra durması	5	8	8	7	6	6	7	168,6504	Politakalar gözden geçirilmeli, kontrol edilmeli
H5		Lokalde uygulamanın çalışması ancak kullanıcı bilgisayarında erişimin sağlanamaması	Ağ bağlantısında oluşan arza veya kesilmesi	8	6	8	6	5	7	7	168,6504	Test ortamında ağ bilgileri kontrol edilmeli

Tablo 5.2.(Devamı) Hata modu etki analizi.

No	Bileşen	Hata	Neden	Olasılık	Farkedilebilirlik	Zaman	Kalite	Maliyet	Çözülebilirlik	Kullanılabilirlik	RÖS	Hafifletme Önerileri
H6	Uygulama Yazılımı	URL adresinde yanlış IP bilgisinin yer alması	Yönlendicinini yanlış yapılveırıması	3	5	3	5	2	6	6	4,840093	Test ortamındaki değişiklerin yönetimi prosedür haline getirilmeli
H7		Uygulamanın kamera erişimine izin vermemesi	Protokol uyuşmazlıı	7	7	5	6	4	6	7	73,78454	Protokollerin kontrol edilmesi. Geliştirmenin bun göre yapılması
H8		Kullanıcılar arası görüşmelerde ses iletiminin gerçekleşmemesi	Chorem yönlendirme protokollerinin izin vermemesi	7	7	8	7	8	6	7	275,4623	Protokollerin kontrol edilmesi. Geliştirmenin bun göre yapılması
H9		Çok fazla işlem peş peşe yapıldığında uygulamanın kullanılmaz hale gelmesi	Yük dağırıcının çalışmaması	5	9	9	7	8	7	8	379,4633	Bakım
H10		Görüntülü ve sesli görüşmelerde bağlantının gidip gelmesi	Kablo kopması veya zayıf sinyal vemesi veri kayıplarına neden olur	8	6	7	8	5	8	8	256,991	Kontrol oluşturulmalı

Tablo 5.2.(Devamı) Hata modu etki analizi.

No	Bileşen	Hata	Neden	Olasılık	Farkedilebilirlik	Zaman	Kalite	Maliyet	Çözülebilirlik	Kullanılabilirlik	RÖS	Hafifletme Önerileri
H11	Uygulama Yazılımı	Uygulamaya giriş parola geçerlilik süresi bitmemesine rağmen uyarı mesajının gösterilmesi	Metot/Kod hatası	3	4	5	7	4	6	7	21,0813	Kod analizi yapılmalı
H12		Uygulamaya entegre edilmiş diğer uygulamara geçişin sağlanamaması	Uygulamanın diğer uygulamlara token vermemesi	7	6	6	7	5	6	7	110,6768	Kod analizi yapılmalı
H13		Profil fotoğraf önizlemesinin gelmemesi	Metot/Kod hatası	9	4	2	5	2	5	5	5,377882	Kod analizi yapılmalı
H14		Kullanıcının pasif olarak kaydedilememesi	Entegre olunan yazılımında ilgili tabloya yazaması	7	8	6	6	4	5	6	72,27873	SQL verileri kontrol edilmeli Yazılım tasarımı kontrol edilmeli
H15		Aktif olan kullanıcının yasaklanmasına rağmen silinmiş gibi gözükmesi	Entegre olunan yazılımında ilgili tabloya yazaması	7	8	6	7	5	6	7	147,5691	SQL verileri kontrol edilmeli YazılımYazılım tasarımı kontrol edilmeli

Tablo 5.2.(Devamı) Hata modu etki analizi.

No	Bileşen	Hata	Neden	Olasılık	Farkedilebilirlik	Zaman	Kalite	Maliyet	Çözülebilirlik	Kullanılabilirlik	RÖS	Hafifletme Önerileri
H16	Uygulama Yazılımı	İlk kullanıcı oluşturma işleminin başarısız ikinci oluşturma işleminin başarılı olması	Veritabanı/Kod hatası	5	7	5	7	5	6	8	87,83873	Database kontrol edilmeli Kod analizi yapılmalı
H17		Başarısız işlem uyarı mesajının alınması	Fazla veri kullanıcımı düşük hızlı disk kullanımı	3	8	1	5	1	4	3	0,430231	Performasnı yüksek disklerin kullanılması
H18		İşlem kayıtları alanında dil özelliklerinin doğru çalışamaması	Metot/Kod hatası	7	5	6	8	6	6	8	144,5575	Kod analizi yapılmalı
H19		Yeni bir kayıt yapılırken ekranın eski veriler ile birlikte açılması	Veritabanında eski verilerin silinmemesi	6	5	5	7	5	6	8	75,29034	Database kontrol edilmeli
H20		Gruplara birden fazla kullanıcının atanamaması	Veritabanı tablosuna erişememesi	2	3	6	7	4	7	8	16,86504	Database kontrol edilmeli

Tablo 5.2.(Devamı) Hata modu etki analizi.

No	Bileşen	Hata	Neden	Olasılık	Farkedilebilirlik	Zaman	Kalite	Maliyet	Çözülebilirlik	Kullanılabilirlik	RÖS	Hafifletme Önerileri
H21	Uygulama Yazılımı	Anahtar geçerlilik süresi alanına 0 değerinin girilmesi	Metot/Kod hatası	9	3	1	3	1	3	3	0,217804	Kod analizi yapılmalı
H22		Ayarlar ekranındaki varsayılan değerlerin doğru gösterilmemesi	Metot/Kod hatası	8	3	1	3	1	2	2	0,086046	Kod analizi yapılmalı
H23		Aynı kullanıcının birden fazla oturum açması	Metot/Kod hatası	6	5	5	7	6	8	7	105,4065	Kod analizi yapılmalı
H24		Parola alanı maskeleme özelliğinin çalışmaması	Metot/Kod hatası	6	5	1	3	1	3	2	0,161336	Kod analizi yapılmalı
H25		Kullanıcı profil alanında kullanıcı bilgilerinin gösterilmemesi	Metot/Kod hatası	8	6	6	7	4	6	7	101,1902	Kod analizi yapılmalı

Tablo 5.2.(Devamı) Hata modu etki analizi.

No	Bileşen	Hata	Neden	Olasılık	Farkedilebilirlik	Zaman	Kalite	Maliyet	Çözülebilirlik	Kullanılabilirlik	RÖS	Hafifletme Önerileri
H26		Aynı eposta adresine sahip birden fazla kullanıcının kaydedilmesi	Metot/Kod hatası	7	5	7	7	7	7	8	200,8579	Kod analizi yapılmalı
H27	Uygulama Yazılımı	Log kayıtlarının oluşturulamaması	Veri tabanı kayıtlarında kullanıcının undefined olarak ataması	6	6	6	7	6	7	6	113,839	Kod analizi yapılmalı. SQL optimizasyonuYazılım tasarımı yeniden gözden geçrilmeli
H28		Güncelenen şifrenin kullanılamaması	Entegre olunan 3.parti yazılımın kendini belli birsüre yenileyememesi	4	6	6	6	6	6	7	65,05086	3.parti aracın şifre değişimini gerçekleştirdiği sürenin incelenmesi. Servisleri yeniden çalıştırmak
H29		Uygulama açılış sayfasının yüklenememesi	Local backedin çalıştırılmaması	5	4	8	7	7	7	9	147,5691	Local backebd exe dosyası kontrol edilmeli

Tablo 5.2.(Devamı) Hata modu etki analizi.

No	Bileşen	Hata	Neden	Olasılık	Farkedilebilirlik	Zaman	Kalite	Maliyet	Çözülebilirlik	Kullanılabilirlik	RÖS	Hafifletme Önerileri
H30		Anahtar geçerlilik süresi alanınan "-" değerlerin girilmesi	Metot/Kod hatası	8	6	1	4	1	3	4	0,688369	Kod analizi yapılmalı
H31		Kullanıcı oluştururken girilen anahtar başlangıç tarihinin kayıt yapıldıkdan sonra yarın başlayacak şekilde göstermesi	Entegre olunan yazılımında ilgili tabloya yazaması	5	7	8	8	6	6	8	192,7433	SQL verileri kontrol edilmeli YazılımYazılım tasarımı kontrol edilmeli
H32	Uygulama Yazılımı	Uygulmadaki kullanıcı görüntü iletiminde kopmaların olması	Veri paketlerinde kayıpların yaşanması	5	8	8	7	6	8	7	224,8672	Test ortamında izole bir ağ ortamın oluşturulması
H33		Belge yükleme butonu yazılarınının okunamaması	Metot/Kod hatası	6	3	2	6	3	3	5	2,904056	Kod analizi yapılmalı
H34		Port bilgisinin sürekli değişmesi	Ağ yapılveırımasını yanlış yapılması	3	8	8	8	6	7	8	154,1946	Test ortamında izole bir ortam oluşturulması

Tablo 5.2.(Devamı) Hata modu etki analizi.

No	Bileşen	Hata	Neden	Olasılık	Farkedilebilirlik	Zaman	Kalite	Maliyet	Çözülebilirlik	Kullanılabilirlik	RÖS	Hafifletme Önerileri
H35		Kullanıcıya yetki verildiğinde uygulamanın kullanılmaz hale gelmesi	Veritabanına erişelememesi	6	8	8	8	7	7	8	359,7874	konfigürasyon Yönetimi, Kod analizi,
H36	Uygulama Yazılımı	İşlem kayıtları alanında listedeki herhangi bir alana basıldığında tüm kayıtların listelenmesi	Metot/Kod hatası	4	8	7	7	7	7	6	137,7311	Kod analizi yapılmalı,Eğitim
H37		Kullanıcı mesajlaşma alanında gönderilen mesajlarıngelen mesajlar alanında gösterilmesi	Metot/Kod hatası	5	7	4	8	4	5	6	40,15485	Kod analizi yapılmalı,Eğitim
H38		Sesli ve görsel uyarı mesajlarının alınamaması	Konfigürasyon hatası	7	5	3	6	4	4	5	15,05807	Kod analizi yapılmalı,Eğitim

Tablo 5.2.(Devamı) Hata modu etki analizi.

No	Bileşen	Hata	Neden	Olasılık	Farkedilebilirlik	Zaman	Kalite	Maliyet	Çözülebilirlik	Kullanılabilirlik	RÖS	Hafifletme Önerileri
H39		Kıdem adı alanına girilen verinin, sayısal değer olarak gösterilmesi	Metot/Kod hatası	3	8	2	5	3	4	6	5,162766	Kod analizi yapılmalı,Eğitim
H40	Uygulama Yazılımı	Silinen anahtar tanımı ile aynı tanımda yeni anahtarın oluşturulaması	Veritabanına erişelememesi	4	6	6	6	5	5	6	38,72075	Kod analizi
H41		Kullanıcının oluşturulamaması	Veritabanı tablosundaki yanlışlık	4	8	8	8	7	8	9	308,3892	Uygulamanın veritabanı tasarımının kontrol edilmesi gerekir
H42		Kullanıcı anahtar bağı koparma işleminin yapılamaması	Metot/Kod hatası	3	5	6	7	4	6	7	31,62194	Kod analizi yapılmalı,Eğitim
H43		Uyarı mesajının yanlış gösterilmesi	Metot/Kod hatası	8	7	1	6	2	2	5	2,007742	Kod analizi yapılmalı,Eğitim

Tablo 5.2.(Devamı) Hata modu etki analizi.

No	Bileşen	Hata	Neden	Olasılık	Farkedilebilirlik	Zaman	Kalite	Maliyet	Çözülebilirlik	Kullanılabilirlik	RÖS	Hafifletme Önerileri
H44		Hesap kilitleme süresi sonrasında doğru şifre ile uygulamaya giriş yapılamaması	Uygulamada entegre çalışan iki uygulama veritabanının uyuşmaması	6	6	7	8	6	6	9	195,1526	Uygulamanın veritabanı tasarımının kontrol edilmesi gerekir
H45	Uygulama Yazılımı	Uygulamada işlem yapılırken uygulamanın, giriş ekranına atması	Sunucu yapılveırımasında cachelerin olması	8	5	8	8	7	7	9	337,3007	Uygulamaya girilmeden önce Cache dosyalarınının kontrol edilmesi ve temizlenmesi
H46		Kullanıcılara atanan görevlerin refreshlemeden gösterilmemesi	Token alamama	8	6	5	6	4	5	6	51,62766	API testi yapılmalı
H47		Logo kullanım hatası	Kaydedilen dosyanın yanlış olması	2	3	1	7	5	5	5	1,568549	Doğru dosya kullanımı sağlanmalı
H48		Versiyon bilgisinin gösterilmemesi	Kod Hatası	7	4	1	3	1	2	1	0,050194	Kod analizi yapılmalı,Eğitim

KAYNAKLAR

Abdullah, A., Saraswat, A. ve Talib, F. (2023). Barriers and strategies for sustainable manufacturing implementation in SMEs: A hybrid fuzzy AHP-TOPSIS framework. *Sustainable Manufacturing and Services Economics*, Vol 2, 9-13

Abdullah, A.G. Shafii, M.A. Pramuditya, S. Setiadipura, T. ve Anzhar K. (2023). Multi-criteria desicion making for nuclear power plant selection using fuzzy AHP: Evidence from Indonesia. *Energy* and AI, Vol14, 3

Abdul, D. Wenqi, J. ve Smeeroddin, M. (2023). Prioritization of ecopreneurship barriers overcoming renewable energy technologies promotion: A comparative analysis of novel spherical fuzzy and pythagorean fuzzy AHP approach. *Technological Forecasting and Social Change*, Vol 186, 1-6

Abusaeed, A. Khan, S.U.R. ve Mashkoor, A. (2023). A fuzzy AHP based approach for prioritization of cost overhead factors in agile software development. *Appplied Soft Computing*, Vol 133, 2-12

Adalı, E.A. ve Işık, A. (2017). Bir tedarikçi problemi için SWARA ve WASPAS yöntemlerine dayanan karar verme yaklaşımı. *International Review of Economics Management*, Vol 5, 56-77.

Agarwal, S. Kant, R. ve Shankar, R. (2020). Evaluating solutions to overcome humanitarian supply chain management barriers: A hybrid fuzzy SWARA – Fuzzy WASPAS approach. *International Journal of Disaster Risk Reduction*, Vol 51, 2-5

Ahsan, F. Naseem, A. Ahmad, Y. ve Sajjad, Z. (2022). Evaluation of manufacturing process in low variety high volume industry with the coupling of cloud model theory and TOPSIS approach. *Quality Engineering*, 35(2), 4-2

Akyuz, E. ve Celik E. (2018). A quantitative risk analysis by using interval type-2 fuzzy FMEA approach: the case of oil spill. *Martime Policy & Management*, Vol 45, 979-994

Alimardani, M. Zolfani A.H. Aghdaie, M.H. ve Tamosaitiene, J. (2013). A novel hybrid SWARA and VIKOR methodology for supplier selection in an agile environment. *Technological and Economic Development of Economy*, Vol 19,533-548

Alvve, A. Mirhosseini, S.M. Ehsanifar, M. Zeighami, E. ve Mohammadi A. (2021). Identification and assessment of risk in construction projects using the integrated FMEA SWARA- WASPAS model under fuzzy environment: a case study of a construction project in Iran., *International Journal of Construction Management*, 23(2), 1-23

Ansari, Z.N. Kant, R. ve Shankar, R. (2020). Evaluation and ranking of solutions to mitigate sustainable remanufacturing supply chain risks: a hybrid fuzzy SWARA-fuzzy COPRAS framework approach. *International Journal of Sustainable Engineering*, Vol 13, 473-495

Butt, S. A. Khalid, A. Ercan, T. Colpas, P.P.A. ve Melisa, A.C. (2022). A software-based cost estimation technique in scrum using a developer's expertise. *Advances in Engineering Software*, 171, 2-3

Calp, M.H. Akcayol M.A. (2015). Yazılım Projelerinde Karşılaşılan Risk Faktörleri ve Risk Yönetim Sürec. *Marmara Fen Bilimleri Dergisi*, 1, 1-4

Cebi, S. Gundugdu, F.K. ve Kahraman, C. Operational risk analysis in business processes using decomposed fuzzy sets, https://content.iospress.com/articles/journal-of-intelligent-ve-fuzzy-systems/ifs213385 adresinden 17 Nisan 2023 tarihinde alınmıştır.

Chen, Y. Hou, Y. Peterson, A. ve Ahmadian, M. (2019). Failure mode and effects analysis of dual levelling valve airspring suspensions on truck Dynamics. *International Journal of Vehicle Mechanics and Mobility*, Vol 57, 617-635

Chen, Y. Shen, S. ve Zhou, A. (2022). Assessment of red tide risk by integrating CRITIC weight method, TOPSIS-ASSETS method, and Monte Carlo simulation. *Environmenal Polution*, Vol 314, 2-4

Chang, D.Y. (1996). Applications of the extent analysis method on fuzzy AHP, *Europen Journal of Operational Research*, Vol 95, 649-655

Chodha, V. Dubey, R. Kumar, R. Singh, S. ve Kaur, S., (2022). Selection of industrial arc welding robot with TOPSIS and Entropy MCDM techniques. *Materialstoday:Proceedings*, Vol 50, 710-712

Çalışkan, D. Yavuz, A.F. Doğan B. Uslu B.Ç. (2021). Türkiye'de Çevik ve Klasik Yazılım Geliştirme Metodolojilerine Dair Kapsamlı Bir Değerlendirme, *BEU Journal of Science*, 10(1), 150-152

Durhan, D. (2006), *Hata Türü Ve Etkileri Analizi (Fmea) Ve Bir Uygulama* [Yüksek Lisans Tezi]. Gazi Üniversitesi

Emovon, I. ve Oghenenyerovwho, O.S. (2020). Application of MCDM method in material selection for optimal design: A review. *Results in Materials*, Vol 7, 2.

Esmaelnezhad, D. Yazdi, M.T. Mahdiraji, H.A. ve Vrontis, D. (2023). International strategic alliance for collaborative product innovation: An agent based scenario analysis in biopharmaceutical indstry. *Journal of Business Research*, Vol 158, 3-8

Erbay, B. (2019). *Bir Yazılım Projesinin Risklerinin Yönetilebilmesi İçin Bulanık Bilişsel Haritalar İle Entegre Edilmiş Bulanık Gri Hata Modu Ve Etkileri Analizi Uygulaması* [Doktora Tezi]. Yıldız Teknik Üniversitesi

Fattahi, R. ve Khalilzadeh, M. (2018). Risk evaluation using a novel hybrid method based on FMEA, extended MULTIMOORA, and AHP methods under fuzzy environment. *Safety Science*, Vol 102, 291-296

Govil, N. ve Sharma A. (2022). Validation of agile methodology as ideal software development process using Fuzzy-TOPSIS method. *Advances in Engineering Software*, Vol 168, 2-5

Gökler, S.H. Yılmaz, D. Uruk, Z.F. ve Boran, S. (2022). A new hybrid risk assessment method based on Fine-Kinney and ANFIS methods for evaluation spatial risks in nursing homes. *Heliyon*, Vol 8, 1-3.

International Software Testing Qualifications Board (ISTQB, Syllbus 2018), Software Test Process https://www.turkishtestingboard.org/files/FL-Syllabus-2018-GA.pdf adresinden 16 Nisan 2023 tarihinde alınmıştır.

İlbahar, E. Kahraman, C. ve Cebi, S. (2022). Risk assessment of renewable energy investments: A modified failure mode and effect analysis based on prospect theory and intuitionistic fuzzy AHP. *Energy*, Vol 239, 1-2

Kersuliene, V. Zavadskas, E.K. ve Turskis, Z. (2010). Selection of rational dispute resolution method by applying new step-wise weight assessment ratio analysis (SWARA). *Journal of Business Economics and Management*, Vol 11, 243–258.

Keshteli, M.H. Cenk, Z. Erdebili, B. Ozdemir, Y.S. ve Jouybari, F.G. (2023). Pythagorean fuzzy TOPSIS method green supplier selection in the food ındustry. *Expert System with Applications*, Vol 224, 2-6

Khalilzadeh, M. Balafshan, R. ve Hafezalkotob, A. (2020), "Multi-objective mathematical model based on fuzzy hybrid multi-criteria decision-making and FMEA approach for the risks of oil and gas projects", *Journal of Engineering Design and Technology*, Vol 1726-0531, 1998-2001

Kılınç, D. Yazılım Yaşam Döngüsü Temel Aşamaları (Software Development Life Cycle Core Processes https://medium.com/@denizkilinc/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-temel-a%C5%9Famalar%C4%B1-software-development-life-cycle-core-processes-197a4b503696 adresinden 5 Nisan 2023 tarihinde alınmıştır.

Koohathongsumrit, N. ve Chankham, W. (2023). Route selection in multimodal supply chains: A fuzzy risk assessment model-BWM-MARCOS framework. *Applied Soft Computing*, Vol 137, 1-7

Korkusuz, P.T. ve Kara, N. (2021). Application of Fuzzy DEMATEL and Fuzzy VIKOR Methods in Personnel Selection. *Europen Journal of Science and Technology*, Vol 23, 377-388

Kumar, C. ve Yadav, D.K. (2015). A Probabilitistic Software Risk A Assesmnet and Estimation Model for Software Projects. *Procedia Computer Science*, Vol 54,354-358

Kumari, S. Ahmad, K. Khan, Z.A. ve Ahmad, S. (2023).Failure mode and effects analysis of common effluent treatment plants of humid sub-tropical regions using fuzzy based MCDM methods. *Engineering Failure Analysis*, Vol 145, 4-6

Kutay, Doruk. (2018). *Importance and Effects Of Contınous Delıvery On Agıle Software Development Lıfecycle* [Yüksek Lisans Tezi]. Yeditepe Üniversitesi

Lee, S.H. Lee, S.J. Koo, S.R. Varuttamaseni, A. Yue, M. Li, M. Cho, J. ve Kang, H.G. (2020). Optimization of software development life cycle quality for NPP safety software based on a risk-cost model". *Annals of Nuclear Energy*, Vol 135, 2-6

Liu, H.C. You, J.X. Ding, X.F. Su, Q. (2015).Improving risk evaluation in FMEA with a hybrid multiple criteria decision making method. *International Journal of Quality & Reliability Management*, Vol 32, 764-770.

Lohakare, P. Bewoor, A. Kumar, R. Said, N.M. ve Sharifpur, M. (2022). Benchmark using multi criteria decision making (MCDM) technique to optimally select piston material. *Engineering Analysis with Boundary Elements*, Vol 142, pp. 55-58

Mardani, A. Nilashi, M. Zakuan, N. Loganathan, N. Soheilirad, S. Saman, M.Z.M. ve Ibrahim, O. (2017). "A systematic review and meta-analysis of SWARA and WASPAS methods: Theory and applications with recent fuzzy developments. *Applied Soft Computing*, Vol 57, 265–292.

Marhavilas, P.K. Filippidis, M. Koulinas, G.K. ve Koulouriotis, D.E. (2020). An expveed HAZOP-study with fuzzy-AHP (XPA-HAZOP technique): Application in a sour crude-oil processing plant. *Safety Science*, Vol 124, 1-4

Moslem, S. Stevic, Z. ve Pilla, F. (2023). Sustainable development solutions of public transportation:An integrated IMF SWARA and Fuzzy Bonferroni operatör. *Sustainable Cities and Society*, Vol 93, 2-5

Moyano, C.G. Pufahl, L. Weber, I. ve Mendling, J. (2022). Uses of business process modeling in agile software development projects. *Information and Software Technology*, Vol 152, 1-3

Naeemah, A.J. ve Wong, K.Y. (2023). Sustainability metrics and a hybrid decision-making model for selecting lean manufacturing tools. Resources, *Environment and Sustainability*, 100120, 11-14

Najihi, S. Elhadi, S. Adelouahid, R.A. ve Marzak, A. (2022). Software Testing an Agile and Traditional View. *Procedia Computer Science*, Vol 203, 777-780

Nath, P. Mushahary, J.R. Roy, U. Brahma, M. ve Singh, P.K. (2023). AI and Blockchain based source code vlnerability detection and prevention system for multiparty software development. *Computer and Electrical Engineering*, Vol 106, 2-10

Nazim, M. Mohammad, C.W. and Sadiq, M. (2022). A comparison between fuzzy AHP and fuzzy TOPSIS methods to software requirements selection. *Alexveeria Engineering Journal*, Vol 6, 10851–10870

Ok, C. Kim, C. Moon, S. Koo, H. Yun, K. ve Lee, S. (2022). Prioritization of radiological accident scenarios during decommissioning of nuclear power plants by risk matrix and AHP method. *Annal of Nuclear Energy*, Vol 175, 2-4

Ouriques, R. Wnuk, K. Gorschek,T. ve Svensson R.B. (2023), "The role of knowledge-based resources in Agile Software Development contexts. *The Journal of System & Software*, Vol 197, 2-7

Pentti, H. ve Atte, H. (2002). Failure mode and Effects Analysis of Software - Based Automation Systems. *STUK - Radiation and Nuclear Safety Authority*, Vol STUK-YTO-TR 190,37.

Pinciroli, F. Justo, J.L.B. ve Forradellas, R. (2022). Systematic mapping study: On the coverage of aspect-oriented methodologies for the early phases of the software development life cycle. *Journal of King Saud University Computer and Information Science*, Vol 34, 2884-2887

Pourmadadkar, M. Beheshtinia, M.A. ve Ghods, K. (2019). An integrated approach for healthcare services risk assessment and quality enhancement. International *Journal of Quality & Reliability Management*, Vol 0265-671X, 763-767.

Pradhan, P. Shabbiruddin, U. ve Pradhan, S. (2022). Selection of electric vehicle using integrated fuzzy-MCDM approach with analysis on challenges faced in hilly terrain. *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects,* Vol 44(2), 2651–2673.

Rabia, M.A.B. ve Bellabdaoui A. (2023).Collabrative intuitionistic fuzzy-AHP to evaluate simulation based analytics for freight transport. *Expert System with Applications*, 26(2), 7-3

Ranjbar, M. Nasiri, M.M. ve Torabi, S.A. (2022). Multi-mode project portfolio selection and scheduling in a build-operate-transfer environment", *Expert Sysem Application*, Vol 189, 1-5.

Rezaee, M. J. Yousefi, S. Eshkevari, M. Valipour, M. ve Saberi, M. (2020). Risk analysis of health, safety and environment in chemical industry integrating linguistic FMEA, fuzzy inference system and fuzzy DEA. *Stochastic Environmental Research and Risk Assessment,* Vol 34(1), 201–218.

Rhmann, W. Pveey, B. Ansari, G. ve Pveey, D.K. (2020).Software fault prediction based on change metrics using hybrid algorithms: An empirical study. *Journal of King Saud University Computer and Information Science,* Vol 32, 420-422

Saeidi, P. Mardani, A. Mishra A.R. 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,* Vol 370, 2-6

Sangwan, O.P. ve Dhvea, M. (2022).A framework for evaluating cloud computing services using AHP and TOPSIS approaches with interval valued spherical fuzzy sets. *Cluster Computing,* Vol 25 (6), 4383–4396

Sarkar, P. Kumar, P. Vishwakarma, D.K. Ashol, A. Elbeltagi, A. Gupta, S. ve Kuriqi, A. (2022). Watershed prioritization using morphometric analysis by MCDM approaches. *Ecological Information,* Vol 70, 3-16

Sarkodie, W.O. Ofosu, E.A. ve Ampimah, B.C. (2022). Decision optimization techniques for evaluating renewable energy resources for power generation in Ghana: MCDM approach. *Energy Reports,* Vol 8,13507-13511.

Seker, S. (2022). IoT based sustainable smart waste management system evaluation using MCDM model under interval-valued q-rung orthopair fuzzy environment. *Technology in Society,* Vol 71, 2-11

Şengül, D. ve Çağıl, G. (2020). Bulanık SWARA ve Bulanık Analitik Hiyerarşi Prosesi Yöntemi ile İş Değerlemesi. *Dicle University Journal of Engineering,* Vol 11, 965-97.

Shafiee, M. ve Animah, I. (2022). An integrated FMEA and MCDA based risk management approach to support life extension of subsea facilities in high-pressure–high-temperature (HPHT) conditions. *Journal of Marina Engineering & Technology,* Vol 21, 189-204

Shaw, D. ve Blundell, N. (2007). WASOP, a qualitative methodology for waste Minimization Systems thinking, HAZOP principles and nuclear waste. *International Journal of Energy Sector Management,* Vol 2, 232-237.

Singh, V. Kumar, V. ve Singh, V.B. (2023). A hybrid novel fuzzy AHP-TOPSIS technique for selecting parameter-influencing testing in software development. *Decision Analytics Journal,* Vol 6, 1-10

Soltan, H. Janad, K. ve Omar, M. (2023). FAQT-2: A Customer-Oriented Method for MCDM with Statistical Verification Applied to Industrial Robot Selection. *Expert Systems with Applications*, Vol 226,9-21.

Souza, F.H. Gaviao, L.O. Sant'Anna, A.P. ve Lima, G.B.A. (2021). Prioritizing risks with composition of probabilistic preferences and weighting of FMEA criteria for fast decision-making in complex scenarios. *International Journal of Managing Project in Business*, Vol 1753-8378, 572-576

Stanujkic, D. Karabasevic, D. ve Karabasevic, E. K. (2015). A framework for the selection ofa packaging design based on the SWARA method. Engineering *Economics*, Vol 26(2), 181–187.

Taheriyoun, M. ve Moradinejad, S. (2015). Reliability analysis of a wastewater treatment plant using fault tree analysis and Monte Carlo simulation. *Environmental Monitoring ve Assessment,* Vol 187 (1), 1-3

Takahashi, M. Ueno, K. Anang, Y. ve Watanable, Y. (2022). A creation method of comprehensive cases and specifications for hardware and software combined test to detect undesirable events of an industrial product using HAZOP. *Realiability Engineering&System Safety*, Vol 15, 50-63

Tomak, N. ve Korkusuz, T. (2022). Risk prioritization model driven by success factor in the light of multicriteria decision making. *Open Chemistry*, Vol 20, 760-764

Ünlükal, C. ve Yücel M., (2021). Risk analysis application ın aviation sector with intuitionistic fuzzy TOPSIS method. *Dumlupınar University Journal of Social Sciences*, Vol 70, 99-101

Verade, J.M.M. Leite, A.F.C.S.M. Cancilieri, M.B. Szejka, A.L. Loures, E.F.R. ve Canciglieri, O. (2022). A multi-criteria decision tool for FMEA in the context of product development and industry 4.0. *International Journal of Computer Integrated Manufacturing*, Vol 35, 36-49

Wang F. ve Gao, J. (2012). A novel knowledge database construction method for operation guidance expert system based on HAZOP analysis and accident analysis. *Journal of Loss Prevention in the Process Industry*, Vol 25, 906-912.

Williams, L. ve Cockburn, A. (2003). Agile Software Development: It's About Feedback and Change. *Institute of Electrical and Electronics Engineers*, 36(6), 39-43

Xie, S. Dong, S. Chen, 0.Y. Peng, Y. ve Li, X. (2021). A novel risk evaluation method for fire and explosion accidents in oil depots using bow-tie analysis and risk matrix analysis method based on cloud model theory. *Realiability Engineering&System Safety*, Vol 215, 3-6

Yener, Y. ve Can, G.F. (2021). A FMEA based novel intuitionistic fuzzy approach proposal: Intuitionistic fuzzy advance MCDM and mathematical modeling integration. *Expert Systems with Application,* Vol 183, 1-3

Yılmaz, Y. (2007), "Bilişim Sistemlerinin Geliştirilmesinde Doğru Yaşam Döngüsü Modelinin Seçimi", *Dergi Park*, Sayı 27, 173-175

Yu, Y. Yang, J. ve Wu, S. (2023). A novel FMEA approach for submarine pipeline risk analysis based on IVIFRN and ExpTODIM-PROMETHEE-I. *Applied Soft Computing* , Vol 136, 2-4

Yücenur, G.N. ve Şenol, K. (2021). Sequential SWARA and fuzzy VIKOR methods in elimination of waste and creation of lean construction processes. *Journal of Bulding Engineering,* Vol 44, 2-6.

Zadeh, L.A. (1968). Fuzzy Algorithms. *Information and Control*, 12(2), 94-102

Zaeri, Ş. Ecer, F. Kostence, D. ve Cheikhrouhou, N. (2021). The vital-immaterial-mediocre multi-criteria decisionmaking method. *Kybernetes,* Vol52, 940-957.

Zhang, K. Duan, M. Luo, X. ve Hou, G. (2017), "A fuzzy risk matrix method and its application to the installation operation of subsea collet connector", *Journal of Loss Prevention in the Process Industries*, Vol 45,147-159

Zhang, Y. ve Xu, L. (2021). Research on risk management of medical and health care integration projects based on fuzzy WINGS-G., *Kybernetes,* Vol 52, 730-743

Zolfani, S.H. Salimi, J. Maknoon, R. ve Kildiene, S. (2015).Technology Foresight about R&D Projects Selection: application of SWARA method at the policy making level. *Engineering Economics*, Vol 26(2),571-580

Zorlu, K. ve Dede, V. (2023). Assessment of glacial geoheritage by multi-criteria decision making (MCDM) methods in the Yalnızçam Mountains, Northeastern Türkiye. *International Journal of Geoheritage and Parks*, Vol 11, 108-110

ÖZGEÇMİŞ

Ad-Soyad : Işılay PAMUK CANDAN

ÖĞRENİM DURUMU:

- **Lisans :** Atılım Üniversitesi, Mühendislik Fakültesi, Endüstri Mühendisliği
- **Yükseklisans :** Devam ediyor, Sakarya Üniversitesi, Endüstri Mühendisliği, Mühendislik Yönetimi

MESLEKİ DENEYİM VE ÖDÜLLER:

- 2021 yılından itibaren ASELSAN iştiraklerinden olan BİTES Savunma şirketinde Yazılım Test Mühendisi olarak çalışmaktayım.
- 2020-2021 yılları arasında BIMCRONE şirkertinde Yazılım Kalite ve Test Sorumlusu olarak çalıştım.