T.C.
İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

YAZILIM TESTİ TASARIM TEKNİKLERİNİN MATEMATİKSEL MODEL YAKLAŞIMI İLE ANALİZİ

Asım Kerem HANCI

Enformatik Anabilim Dalı

Enformatik Programı

DANIŞMAN
Yrd. Doç. Dr.İnci ZAİM GÖKBAY

Aralık, 2017

İSTANBUL

Bu çalışma 01.12.2017 tarihinde aşağıdaki jüri tarafından Enformatik Anabilim Dalı, Enformatik Programında Yüksek Lisans tezi olarak kabul edilmiştir.

Tez Jürisi

Yrd. Doç. Dr. İnci ZAİM GÖKBAY(Danışman)
İstanbul Üniversitesi
Fen Fakültesi

Prof. Dr. Sevinç GÜLSEÇEN
İstanbul Üniversitesi
Fen Fakültesi

Prof. Dr. Ayla ŞAYLI
Yıldız Teknik Üniversitesi
Kimya Metalurji Fakültesi

Doç. Dr. Sedat BALLIKAYA
İstanbul Üniversitesi
Mühendislik Fakültesi

Yrd. Doç. Dr. Tarık Veli MUMCU
İstanbul Üniversitesi
Mühendislik Fakültesi

20.04.2016 tarihli resmi gazetede yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, İstanbul Üniversitesi'nin abonesi olduğu intihal yazılım programı kullanılarak Fen Bilimleri Enstitüsü'nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

ÖNSÖZ

‘Yazılım Testi Tasarım Tekniklerinin Matematiksel Model Yaklaşımı ile Analizi’ adlı bu çalışma İ.Ü. Enformatik A.B.D.’de yüksek lisans bitirme tezi olarak yapılmıştır.

Bitirme tezimin konusunun belirlenmesinde, planlanmasında ve çalışmalarımın yürütülmesinde bana destek olan tez danışmanım Sayın Yrd. Doç. Dr. İnci ZAİM GÖKBAY’ a teşekkürlerimi sunarım.

Tez sürecim ve öncesinde bana her türlü destek olan ve fikirlerimize değer verip değerlendirmemiz için gerekli ortamı sağlayan bölüm başkanımız Sayın Prof. Dr. Sevinç GÜLSEÇEN’ e ve diğer öğretim üyelerine sonsuz teşekkürlerimi sunarım.

Tez çalışmamın fikrinin oluşmasında ve tez ile ilgili veri toplamam için benden desteklerini esirgemeyen Barış SARIALİOĞLU’ na; akademik olarak yola devam etmemi teşvik eden Cem BURUŞ’ a; bana özgür bir çalışma ortamı sağlayarak çalışmalarıma devam etmemi kolaylaştıran Zehra TAŞGIN’ a ve Derya ÇETİN’ e sonsuz teşekkürlerimi sunarım.

Son olarak, benden maddi ve manevi desteklerini hiçbir zaman esirgemeyen anneme teşekkürlerimi sunarım.

Aralık, 2017 Asım Kerem HANCI

İÇİNDEKİLER

Sayfa No

ÖNSÖZ ..... **iv**
İÇİNDEKİLER ..... **v**
ŞEKİL LİSTESİ ..... **vii**
TABLO LİSTESİ ..... **viii**
SİMGE VE KISALTMA LİSTESİ ..... **x**
ÖZET ..... **xi**
SUMMARY ..... **xii**
1. GİRİŞ **1**
2. GENEL KISIMLAR **3**
2.1.YAZILIM TESTİ 3
2.1.1.Testin Yedi Prensibi 4
2.1.2.Test Süreçleri 6
2.1.3.Test Seviyeleri 11
2.1.4.Test Çeşitleri 13
2.1.5.Risk 15
2.1.6. Hata Yönetimi 16
2.2.TEST TASARIM TEKNİKLERİ 18
2.2.1.Statik Test 18
2.2.2.Dinamik Test 21
3. MALZEME VE YÖNTEM **38**
3.1.TEST TASARIM TEKNİKLERİ KULLANIM SIKLIĞI ANALİZİ 38
3.1.1.Anket Soruları Analizi 38
3.1.2.Anket Sonuçları Analizi 45
3.2.TEST TASARIM TEKNİKLERİ HATA BULMA ORANLARI ANALİZİ 57
3.2.1.Gereksinim Bazlı Test Tekniklerinin Hata Bulma Oranları Açısından Sıralaması 57
3.2.2.Yapı Bazlı Test Tekniklerinin Hata Bulma Oranları Açısından Sıralaması 59
3.2.3.Deneyime Dayalı Test Tekniklerinin Hata Bulma Oranları Açısından Sıralaması 59

3.2.4.Statik Test Tekniklerinin Hata Bulma Oranları Açısından Sıralaması 61
3.3.TEST TASARIM TEKNİKLERİ UYGULANABİLİRLİK ANALİZİ 63
3.3.1.Gereksinim Bazlı Test Tekniklerinin Uygulanabilirlik Açısından Sıralaması 63
3.3.2.Yapı Bazlı Test Tekniklerinin Uygulanabilirlik Açısından Sıralaması 65
3.3.3.Deneyime Dayalı Test Tekniklerinin Uygulanabilirlik Açısından Sıralaması 66
3.3.4.Statik Test Tekniklerinin Uygulanabilirlik Açısından Sıralaması 67
3.4.TEST TASARIM TEKNİKLERİNİN MATEMATİKSEL MODELLEMESİ.....70
4. BULGULAR 73
4.1.GEREKSİNİM BAZLI TEST TEKNİKLERİ İÇİN KARAR VERME 73
4.2.YAPI BAZLI TEST TEKNİKLERİ İÇİN KARAR VERME.....75
4.3.DENEYİME DAYALI TEST TEKNİKLERİ İÇİN KARAR VERME 76
4.4.STATİK TEST TEKNİKLERİ İÇİN KARAR VERME 77
5. TARTIŞMA VE SONUÇ 79
KAYNAKLAR 80
ÖZGEÇMİŞ.....83

ŞEKİL LİSTESİ

Sayfa No

Şekil 2.1: Yazılım Testi Süreçleri. 6

Şekil 2.2: Denklik Paylarına Ayırma.....22

Şekil 2.3: Zayıf Normal Denklik Payı Grafiği 24

Şekil 2.4: Sağlam Normal Denklik Payı Grafiği 24

Şekil 2.5: Zayıf Kapsamlı Denklik Payı Grafiği. 25

Şekil 2.6: Sağlam Kapsamlı Denklik Payı Grafiği. 26

Şekil 2.7: Sınır Değer Analizi. 27

Şekil 2.8: Zayıf Normal Sınır Değer Analizi.....28

Şekil 2.9: Sağlam Normal Sınır Değer Analizi 28

Şekil 2.10: Zayıf Kapsamlı Sınır Değer Analizi 29

Şekil 2.11: Sağlam Kapsamlı Sınır Değer Analizi 29

Şekil 2.12: Durum Geçiş Diyagramı. 31

Şekil 2.13: Kullanım Senaryosu. 33

Şekil 2.14: Komut ve Karar Testi. 34

Şekil 3.1: Yazılım Testi Çalışanlarının Deneyim Kümesi. 39

TABLO LİSTESİ

Sayfa No

Tablo 2.1: Karar Tablosu. 30

Tablo 3.1: Gereksinim Bazlı Test Tasarım Tekniklerinin Hata Bulma Oranları Açısından Tercih Tablosu. 58

Tablo 3.2: Deneyime Dayalı Test Tasarım Tekniklerinin Hata Bulma Oranları Açısından Tercih Tablosu. 60

Tablo 3.3: Statik Test Tasarım Tekniklerinin Hata Bulma Oranları Açısından Tercih Tablosu. 62

Tablo 3.4: Gereksinim Bazlı Test Tasarım Tekniklerinin Uygulanabilirlik Açısından Tercih Tablosu. 65

Tablo 3.5: Deneyime Dayalı Test Tasarım Tekniklerinin Uygulanabilirlik Açısından Tercih Tablosu. 67

Tablo 3.6 : Statik Test Tasarım Tekniklerinin Uygulanabilirlik Açısından Tercih Tablosu. 69

Tablo 4.1: Gereksinim Bazlı Test Tasarım Teknikleri Parametre Değer Tablosu. 73

Tablo 4.2: Gereksinim Bazlı Test Tasarım Teknikleri Parametre Normalize Değer Tablosu. 74

Tablo 4.3: Gereksinim Bazlı Test Tasarım Teknikleri Karar Değer Tablosu. 74

Tablo 4.4: Yapı Bazlı Test Tasarım Teknikleri Parametre Değer Tablosu. 75

Tablo 4.5: Yapı Bazlı Test Tasarım Teknikleri Parametre Normalize Değer Tablosu. 75

Tablo 4.6: Yapı Bazlı Test Tasarım Teknikleri Karar Değer Tablosu. 76

Tablo 4.7: Deneyime Dayalı Test Tasarım Teknikleri Parametre Değer Tablosu. 76

Tablo 4.8: Deneyime Dayalı Test Tasarım Teknikleri Parametre Normalize Değer Tablosu. 76

Tablo 4.9: Deneyime Dayalı Test Tasarım Teknikleri Karar Değer Tablosu. 77

Tablo 4.10: Statik Test Tasarım Teknikleri Parametre Değer Tablosu. 77

Tablo 4.11: Statik Test Tasarım Teknikleri Parametre Normalize Değer Tablosu 78

Tablo 4.12: Statik Test Tasarım Teknikleri Karar Değer Tablosu. 78

SİMGE VE KISALTMA LİSTESİ

Simgeler	**Açıklama**
X	: Rastgele değişken
P(X)	: X olaylarının olasılığı
P(X|X_k)	: X_k olayının koşullu olasılığı
F(X)	: X olayının karar fonksiyonu

Kısaltmalar	**Açıklama**
ISTQB	: International Software Testing Qualifications Board
TTB	: Turkish Testing Board
TDK	: Türk Dil Kurumu

ÖZET

Yazılım Testi Tasarım Tekniklerinin Matematiksel Model Yaklaşımı ile Analizi

YÜKSEK LİSANS TEZİ

Asım Kerem HANCI

İstanbul Üniversitesi

Fen Bilimleri Enstitüsü

Enformatik Anabilim Dalı

Danışman: Yrd. Doç. Dr. İnci ZAİM GÖKBAY

Teknolojinin hayatın her alanına girmesiyle kullanılmaya başlanan yazılımın kaliteli olması son yıllarda oldukça önem kazanmıştır. Yazılımın kaliteli olması ihtiyacı yazılım geliştirme yaşam döngüsünde test sürecinin yapılanmasını beraberinde getirmiştir. Yazılımın kaliteli olması adına gerçekleştirilen testlerde farklı tasarım teknikleri kullanılmaktadır. Bu çalışma; yazılım testi tasarım tekniklerinin matematiksel modellemesini ve bu modellemenin analizini içermektedir. Çalışma sırasında yazılım testi tasarım teknikleri ayrıntılı olarak tanımlanmış, her bir teknik kullanım sıklığı, hata bulma oranı ve uygulanabilirlik açısından değerlendirilmiştir. Bu değerlendirme ölçütleri matematiksel modelde kullanılarak yazılım testi için en uygun olan teknikler tespit edilmiştir. Yazılım testi tasarım teknikleri matematiksel modellemesinde Bayes Teoremi ve vaka analizinden faydalanılmıştır.

Aralık 2017, 83 sayfa.

Anahtar kelimeler: Yazılım testi, test tasarım teknikleri, yazılım testi modelleme

SUMMARY

Analysis of Software Test Design Techniques by Mathematical Modelling

M.Sc. THESIS

Asım Kerem HANCI

İstanbul University

Institute of Graduate Studies inScience and Engineering

Department of Informatics

Supervisor : Asst. Prof. Dr. İnci ZAİM GÖKBAY

It has come into prominance in recent years that softwares, which started to be used as technologyentered in every aspect of life, be of good quality. The need for the software to be qualified brought the making of testing process in the software development lifecycle along. Various design techniques are used in the conducted tests in order that the software test design techniques and the analysis of this modelling. Software test design techniques are detailly described and each technique are evaluated in terms of frequency of use, defect detection rate and applicability. The optimal techniques for software testing are determined by using these evaluation criteria in the mathematical modelling. Bayes' theorem and case analysis are profited in the mathematical modelling of the software test design techniques.

December 2017,83 pages.

Keywords: Sotware testing, test design techniques, software test modelling

1. GİRİŞ

Bilim Çağı olarak adlandırılan günümüzde bilim ve bilimin uygulanması sonucu olarak ortaya çıkan teknoloji; savunmadan sağlığa, eğitimden ulaşıma kadar hayatımızın her alanında yer alarak insanoğlunun vazgeçilmezleri arasına girmeyi başarmıştır. Teknoloji, bilgisayarın veya diğer iletişim araçlarının kullanılmasıyla oluşturulmuş sistemlerin geliştirilmesini kapsar. Sistem geliştirmeleri, ufak değişiklikler olabileceği gibi teknolojide kilometre taşı sayılabilecek köklü değişiklikler de olabilir. Yapılan bütün bu geliştirmeler yazılımdan başlayarak gerçekleştirilir. Yazılım, değişik ve çeşitli görevler yapma amaçlı tasarlanmış elektronik aygıtların birbiriyle iletişimini sağlayan makina komutlarıdır [1].

Bir yazılım ürünü, birkaç aşamadan oluşan bir sürecin tamamlanmasıyla son şeklini alır ve kullanıma hazır hale gelir. Bu sürece *yazılım geliştirme yaşam döngüsü* adı verilir. Yazılım ile ilgili gereksinimler sürekli olarak değişebilir nitelikte olduğundan aşamalardan oluşan bu süreç döngü olarak adlandırılmıştır. Yazılım geliştirme yaşam döngüsü temel olarak aşağıdaki aşamalardan oluşur [2]:

- Planlama
- Analiz
- Tasarım
- Kodlama
- Test
- Bakım

Planlama aşaması; ihtiyaçların belirlendiği, proje planının oluşturulduğu ve fizibilite çalışmasının yapıldığı aşamadır [2]. Zaman ve kaynak ayrımı planlanması, personelin ve gereksinimlerin belirlenmesi bu aşamada gerçekleşir.

Analiz aşaması; belirlenen gereksinimlerin ve mevcut sistemin tahlil edildiği aşamadır. Mevcut sistem ve yeni sistemin kıyaslanıp optimize kararlar alınarak sistem gereksinimleri ve işlevleri ayrıntılı olarak ortaya çıkarılır [3]. Talepler net olarak

hazırlanır. Analiz edilen bütün nitelikler analiz dokümanlarında belirtilir. Analiz dokümanları yazılım geliştirme yaşam döngüsünün bundan sonraki aşamaları için rehber niteliği taşır. Çünkü yapılan bütün geliştirmeler analiz dokümanlarındaki gereksinim ve modellemelere göre gerçekleştirilir.

Tasarım aşaması; analizi yapılmış olan gereksinimlere cevap verebilecek nitelikte bir sistemin mimarisinin oluşturulduğu aşamadır. İşlem adımları, kullanıcı ara yüzleri, algoritmalar bu aşamada oluşturulur. Tasarımı oluşturulan kısımlar için tasarım dokümanları hazırlanır. Tasarım dokümanı yazılımcıların kodunu yazarken referans alacağı veri modeli, iş akışları ve ayrıntılı tasarım bilgilerini içerir [2].

Kodlama aşaması; bu zamana kadar yapılan bütün işlemlerin ürüne dönüştürüldüğü aşamadır. Kodlamanın yanında kurulum, kod yönetimi, birim test gibi işlemler de bu aşamada gerçekleşir.

Test aşaması, oluşturulan ürünün kullanıcıya sunulmadan önce gereksinimleri karşılama derecesinin belirlendiği aşamadır. Hatalar tespit edilir; performans, kullanılabilirlik standartlarına göre ürünün uygunluğu belirlenir. Bu aşamada test edilen ürün yalnızca kod değildir. Analiz dokümanı, tasarım dokümanı gibi diğer proje çıktıları da test edilecek ürünler arasındadır.

Bakım aşaması, test sırasında tespit edilen hataların giderildiği ve gereksinimlere uygun hale getirildiği son aşamadır. Projenin yapısı değerlendirilir ve bu bilgiler arşivlenir.

Bu tez çalışmasında; yazılım geliştirme yaşam döngüsünün beşinci aşaması olan test aşamasının tasarım teknikleri ayrıntılı olarak ele alınmıştır. Türkiye'de gerçekleştirilen yazılım projelerinde kullanılan test tasarım teknikleri araştırılmış ve kullanım sıklığı, hata bulma oranı ve uygulanabilirlik açısından değerlendirilerek test süreç analizi yapılmıştır. Analiz çalışmalarında Türkiye'nin ileri gelen firma ve kuruluşlarında görev alan yazılım testi profesyonellerinin düşüncelerinden faydalanılmıştır.

2. GENEL KISIMLAR

2.1.YAZILIM TESTİ

Günümüzde teknolojinin ilerlemesini sağlayan en önemli etkenlerden birisi bilişim sistemleridir. Bilişim sitemleri yazılım tabanlı olup yazılım kullanılarak oluşturulur veya geliştirilir. Yazılım; bir sistemi belirli işlevleri yerine getirmek üzere yöneten, sisteme ne yapacağını söyleyen, kodlanmış komutlar dizisi olarak tanımlanabilir. Yazılım hayatımızın birçok alanında kullanıldığından kaliteli olması hem bilgi güvenliği hem de para, zaman, itibar kaybı gibi problemlerin yaşanmaması için oldukça önemlidir.

Kaliteli yazılımlar; gereksinimleri karşılayabilecek nitelikte, kabul edilebilir düzeyde hata payına sahip, planlanan bütçeyle öngörülen zamanda bitirilebilen ve sürdürülebilir olan yazılımlardır [4]. Kalite öznel bir kavram gibi görünse de kaliteyi ortaya koyan nesnel yöntemler yansız değerlendirmeleri olanaklı kılmaktadır. Beklenen ölçüde kaliteli ve hatasız çalışan yazılımların üretilmesi için yazılım basamaklarının bütününde yazılım kalite kontrolü -başka bir deyişle- yazılım testi uygulanmalıdır.

Test; geliştirilen ürün kalitesi hakkında paydaşlara bilgi sağlamak amacıyla ürünün sahip olduğu sistem için belirlenmiş gereksinimlerin karşılandığının doğrulanması ve karşılanmaması durumunda beklenen ile gözlenen durumların arasındaki farkların belirlenmesini kapsayan bir araştırma sürecidir. Yazılım testi; yazılım uygulamalarının risklerini anlamak için sonsuz sayıdaki çalışma alanından sınırlı sayıda ve uygun şekilde seçilmiş senaryolar ile gereksinimleri karşılamaya yönelik bağımsız olarak inceleme faaliyetlerini kapsar [5]. ISTQB(International Software Testing Qualifications Board) temel seviye müfredatında belirtildiği üzere yazılım testi genel olarak;

- Hataları bulma,
- Kalite seviyesi hakkında güven kazanma,
- Karar verme için bilgi sağlama,
- Hataları önleme amaçları doğrultusunda yapılmaktadır [6].

Yazılım geliştirme yaşam döngüsünün başlangıcında belirlenen gereksinimler baz alınarak geliştirilen yazılımlarda, gelişen teknolojiye rağmen hataların olması kaçınılmazdır. Geliştirilen yazılımın uygulama alanına göre hatanın kritikliği değişebilmektedir. Yazılımın kullanılacağı alana ya da uygulamanın kullanıcılarının uygulamayı kullanarak yerine getirdikleri çalışmaların türüne göre ekonomiden sağlığa, ülke güvenliğinden şirket güvenliğine kadar pek çok farklı alanda önemli sorunlar oluşmasında etken olabilir. Örneğin; bir bankacılık uygulamasında gözden kaçırılan ufak bir güvenlik açığının ilgili banka için büyük ölçüde itibar ve para kaybına yol açabileceği gibi sağlık alanında geliştirilen bir uygulamadan kaynaklanan hata can kaybına sebep olabilmektedir. American National Institute of Standards and Technology (NIST)‘ nin 2002 raporu, Birleşik Devletler‘ de yazılım test altyapısının olmamasının tek başına negatif etkisinin yıllık 62 Milyar Dolar olduğunu raporlamıştır [7]. Diğer ülkelerde de benzer tablonun olduğunu tahmin etmek mümkündür. Bu nedenle yazılımın test edilmesi ve güvenilir ölçüde çalıştığının kanıtlanması çok büyük önem arz etmektedir.

2.1.1.Testin Yedi Prensibi

Yazılım testleri gerçekleştirilirken farklı bakış açılarından faydalanılmıştır. ISTQB’ nin temel seviye müfredatında üzerinde durduğu aşağıda bahsedilen prensipler; bu bakış açılarının ortak özelliklerini özetlemektedir [8]:

Prensip 1: Testin amacı, yazılımda hataların olduğunu göstermektir; yazılımda hata kalmadığını ispatlamak değildir.

Yazılımlar; hataların tespit edilip mümkün olduğunda da düzeltilmesi için test edilir. Hata kalmadığını ispatlamak için yapılan testler yanlış bakış açısıyla test edilmiş olur. Test aktiviteleri gerçekleştirilirken test uzmanı sisteme pesimist bakış açısıyla yaklaşmalı, sistemin eksik yönlerini bulma amacıyla testleri koşmalıdır.

Prensip 2: Yazılımı %100 test etmek imkansızdır.

Bir yazılım ürününün tümüyle test etmek mümkün olmaz çünkü proje için ayrılan zaman ve bütçe buna izin vermez. Bu nedenle riskli alanlar tespit edilip önceliklendirilerek zamanın ve bütçenin elverişliliği ölçüsünde test aktiviteleri gerçekleştirilir.

Prensip 3: Teste yazılım geliştirme sürecinin başında başlamak gerekir.

Teste yazılım geliştirme yaşam döngüsünün erken aşamalarında başlanması; projenin hata maliyetinin azaltılması için oldukça önemlidir. Testin sürece erkenden dahil olmasıyla birçok hata sisteme girildiği aşamada tespit edilerek maliyet en aza indirilmiş olur.

Prensip 4: Hatalar yazılımın belli alanlarında yoğunlaşır.

Test esnasında bulunan hatalar, aynı modüldeki başka hataların habercisidir. Test uzmanları hata tespit ettikleri modülleri daha kapsamlı test etmelidir. Hataların %80'lik bir kısmının yazılımın %20'lik bir kısmından kaynaklandığı; birçok projede kanıtlanmış bir gerçektir.

Prensip 5: Antibiyotik Direnci

Bir yazılım sürekli aynı test senaryolarıyla ve aynı tekniklerle test edilirse bir süre sonra o yazılımda hata alınmamaya başlanır. Çünkü bulunan hatalar sonucunda yazılım direnç kazanmaya başlamıştır. Bundan dolayı test senaryolarının periyodik olarak güncellenmesi, yazılımın daha sağlıklı test edilmesini sağlar.

Prensip 6: Test yaklaşımı yazılım projelerinin koşullarına göre değişiklik gösterir.

Test içerik bağımlı bir aktivitedir. Yazılımın özelliklerine, bağlamına ve içeriğine göre farklı biçimlerde ele alınmalıdır. Örneğin; bir bankacılık uygulaması ile bir medikal sistem için yazılmış bir yazılım farklı güvenlik seviyelerinde ve farklı tekniklerle test edilmelidir.

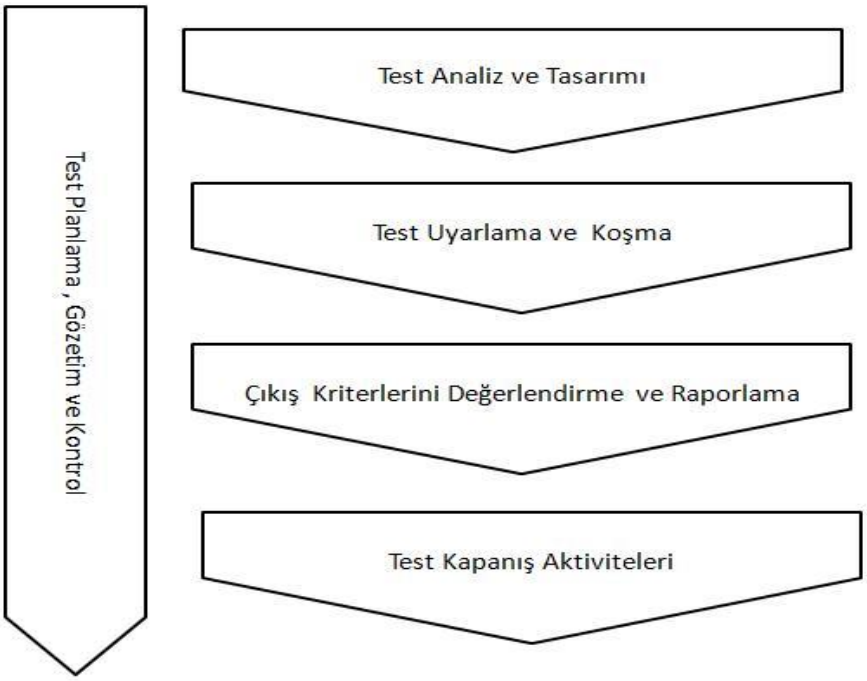
Prensip 7:"Yeni hata bulamıyoruz, başarılı bir yazılım elde ettik." Yanılgısı

Test uzmanları tarafından tespit edilen hatalar düzeltildikten sonra yeni hataların çıkabileceği veya müşterinin ihtiyaçlarının tam anlamıyla karşılanamayabileceği riskleri unutulmamalıdır. Bir yazılımın doğru çalışıyor olması onun mutlak doğru olduğunu göstermez. Gereksinimler farklı geliştirme ihtiyacı doğurabilir.

2.1.2.Test Süreçleri

Yazılım testi, yazılım geliştirme yaşam döngüsü ile paralel olarak ilerleyen bir dizi aşamadan oluşan bir süreçten meydana gelmektedir. Bu aşamalar aşağıdaki gibi özetlenebilir:

- Test Planlama, Gözetim ve Kontrol
- Test Analizi ve Tasarımı
- Test Uyarlama ve Koşma
- Çıkış Kriterlerini Değerlendirme ve Raporlama
- Test Kapanış Aktiviteleri



Şekil 2.1: Yazılım Testi Süreçleri**.**

2.1.2.1.Test Planlama, Gözetim ve Kontrol

a. Test Planlama

Test planlama; test sürecinin başından başlanarak test faaliyetlerinin tümü tamamlanana kadar devam eden bir safhadır. Test strateji dokümanlarında tanımlanmış olan test hedeflerinin gerçekleştirilmesi için gereken test faaliyetlerinin planlaması bu aşamada gerçekleşir. Test planlama aşamasında ayrıca projeye kılavuzluk eden

metrikler tanımlanır. Bu metrikler ışığında test planlaması yapılırken gerek test araçları seçilebilir, gerekse eğitimler organize edilebilir [9].

Test planlama aktiviteleri genellikle test yöneticileri ve ona bağlı test analistleri ve teknik test analistleri tarafından gerçekleştirilir. Belirlenen test yaklaşımına uygun olarak test tahmini yapılır ve bu bilgilere göre eforlar belirlenir. Örneğin; mobil uygulamalar için kullanılabilirlik önemli bir kalite unsuru olduğundan kullanılabilirlik testlerine ayrılan zamanın yeterli olduğundan emin olunmalıdır.

Test planlama aşamasında yaklaşımlar ve riskler açıkça belirlenir. Riskler belirlenirken paydaşlar mutlaka dahil edilmelidir. Çünkü her paydaşın kendi açısından belirleyeceği riskler farklı olabilir. Örneğin iş birimleri müşteriye yansıyacak hataların riskini belirlerken yazılım uzmanları teknik riskleri tanımlayabilir.

Test planlanırken kapsamda olan ve olmayan özellikler de açıkça belirtilmelidir. Yazılımın hangi parçalarının test edilip hangi parçalarının test edilmeyeceği test kapsamının net anlaşılması açısından büyük önem taşır.

b. Test Gözetimi ve Kontrol

Test planlama aşaması gibi test gözetimi ve kontrol aşaması da test sürecinin başından sonuna kadar devam eder. Test sürecinin test planına ne kadar uygun gittiği kontrol edilir. Plandan sapmalar tespit edildiğinde nedenleri araştırılır ve yeni test planı oluşturulur.

Test kontrolü için metrikler büyük önem taşır. Plana uygunluk veya plandan sapmalar; test süreci boyunca toplanan metriklerle belirlenir. Metriklerin gerçek bilgileri objektif bir şekilde yansıtması alınacak aksiyonlar açısından oldukça önemlidir. Yöneticiler bu metriklerden faydalanarak test sürecine müdahale eder.

Metrikler toplanırken zamanlama da göz önünde bulundurulmalıdır. Doğru metriklerin doğru zamanlarda tespit edilmesi ve paydaşlarla paylaşılması yapılacak değişikliklerin zaman kaybetmeden gerçekleştirilmesini sağlar.

2.1.2.2.Test Analizi ve Tasarımı

a. Test Analizi

Test analizi; test koşullarının tanımlandığı ve *test esası* adı verilen gereksinimlerin çıkarılabileceği belgelerin analiz edildiği aşamadır. Bu aşamada test koşullarına dayanılarak test edilecek kısımlar net olarak belirlenir. Test koşulları da test esasına, test hedeflerine ve ürün risklerinin analiz edilmesi ile ortaya çıkar. Test koşullarının belirlenmesi birçok parametreye bağlıdır (test seviyesi, yazılımın kompleksliği, yazılım geliştirme yaşam döngüsü modeli gibi) [10].

Test koşularının tanımlanma detayı projenin kritikliğine göre değişebilir. Savunma sanayi uygulamaları veya sağlık sektöründeki geliştirmelerde test koşulları tanımlanırken test analizi aşaması daha detaylı ve daha fazla dokümantasyon gerektiren aşamalar içerir. Gereksinimler en ayrıntılı biçimde gözden geçirilir, mümkün olan en küçük parçalara ayrılarak bunları kapsayan test koşulları tanımlanır. Böylece yazılım canlı ortama alındığında hata oluşma riski en aza indirilir.

b. Test Tasarımı

Test tasarımı; test analizi aşamasında tanımlanan test koşullarını karşılayan test senaryolarının oluşturulduğu ve tasarım tekniklerinin belirlendiği aşamadır. Bu aşamada yazılımın nasıl test edileceği net olarak belirlenir.

ISTQB'nin ileri seviye müfredatına göre senaryolar iki başlık altında sınıflandırılır [11]:

- **Somut Test Senaryoları:** Gereksinimlerin açık bir şekilde tanımlandığı senaryolardır. Kusursuz bir şekilde tekrar edilebilir.
- **Mantıksal Test Senaryoları:** Test koşulurken gereksinimlerin kapsamı genişletilerek oluşturulan senaryolardır. Testi koşan kişinin deneyimine bağlı olarak detayı değişebilir.

Test senaryoları oluşturulurken hedeflerin ve beklenen sonuçların test analizinde belirlenen koşullar ile uyumlu olmasına dikkat edilmelidir. Senaryoların detay seviyesi analiz aşamasındaki gibi projenin kritikliğine göre değişebilir. Test senaryolarının

detaylı olması kapsamı artırarak hata riskini azaltırken; az detay ile hazırlanmış senaryolar test uzmanına testi koşarken esneklik sağlar.

Test senaryoları hazırlanırken proje ve ürün riskleri, seçilen yazılım geliştirme yaşam döngüsü, kapsam, zaman, personel gibi birçok parametre göz önünde bulundurulmalıdır. Her döngü ile her tasarım tekniği uyumlu olmayabilir. Örneğin şelale modelinde kodlama bitirilmeden önce test aktivitelerine başlanamayacağından şelale modelinde kodlamadan önce gerçekleştirilen statik test teknikleri uygulanamaz. Benzer şekilde test aktivitelerine az bütçenin ayrıldığı projelerde karar tablosu gibi detaylı çalışma gerektiren test tekniklerinin kullanılması çok doğru olmaz.

Test senaryolarının bütün paydaşlar tarafından açıkça anlaşılacak şekilde oluşturulmuş olması gerekir. Çünkü oluşturulan test senaryoları risk kapsamı belirlenmesi sırasında kritik bir önem taşır. Ayrıca test senaryolarının sadece senaryoyu hazırlayan kişi tarafından değil diğer test uzmanları tarafından da koşulabilmesi sağlanmış olur.

2.1.2.3.Test Uyarlama ve Koşma

a. Test Uyarlama

Test uyarlama; test adımlarının belirlendiği ve test koşumu için gerekli hazırlıkların tamamlandığı aşamadır. Test uyarlama aşamasında somut test senaryoları nihai haline getirilir, test verileri oluşturulur ve test edilecek ortamın test koşumu için uygun olduğu kontrol edilir.

Test adımları; test senaryolarını oluşturulan yönergeler ve beklenen sonuçlardan oluşan test yapı birimleridir. Bir test senaryosu tek adımdan oluşabileceği gibi çok sayıda adım içeren test senaryoları oluşturmak da mümkündür. Oluşturulan test adımları test senaryosunu detaylandıran, doğru koşulu sağlayan nitelikte olmalıdır.

Test verileri test yaparken kullanılan işlenmiş bilgileri içerir. Senaryoların doğru veri ile koşulması oldukça önemlidir. Çünkü yanlış verilerle gerçekleştirilen testler hata olmayan yerlerde hata oluşmasına (false-positive) veya hata olan yerlerde hatanın tespit edilememesine (false-negative) sebep olur [12].

Test ortamının hazır olması test koşumu için en kritik etkendir. Test koşulamayan ortam zamanın uzamasına ve plandan sapmalara sebep olur. Test koşumu sırasında dikkate alınması gereken çok sayıda unsur olacağından ortamın elverişliliğinin teyit edilmesi gerekir.

b. Test Koşumu

Test koşumu; test hazırlıkları tamamlandıktan sonra test senaryolarının koşulduğu aşamadır. Koşum sırasında gerçekleşen sonuçlarla test adımlarındaki beklenen sonuçlar karşılaştırılır.

Analiz, tasarım ve uyarlama aşamalarında oluşturulan test ürünlerine göre test koşumu yapılabileceği gibi; mantıksal test senaryolarının da koşulması için yeterli zaman ayrılmalıdır. Böylece yazılım daha geniş kapsamda test edilmiş olur.

Beklenen sonuçlar ve gerçekleşen sonuçlar birbirinden farklı ise bu duruma *anomali* denir. Anomali meydana geldiği durumda test dokümanları tekrar incelenir. Gerçekleşen sonuç test dokümanlarında belirlenen koşulları karşılamıyorsa hatanın tespit edildiği anlaşılır. Tespit edilen hata doğru, objektif ve anlaşılır bir şekilde kayıt altına alınmalıdır. Böylece hata raporunu üstlenen yazılım uzmanı hata ile ilgili daha fazla bilgiye ihtiyaç duymaz.

Test koşumu sırasında yazılımın yapması gereken davranışları yapmasının yanında; yapmaması gereken davranışları yapmadığına dikkat edilmelidir [13]. Bu gözlem test koşumu sırasında sıklıkla atlanan bir konu olmakla birlikte yazılımın canlıya alınmasında karşılaşılacak hataların en aza indirilmesi için faydalı bir yöntemdir.

Test koşumu sırasında test araçlarının kullanılması; hem test sürecinin hem de hata raporlarının takibi açısından oldukça önemlidir. Test araçlarından elde edilen bilgilere göre paydaşlar test koşum oranı ve tespit edilen hatalar hakkında daha net bilgiye sahip olur.

2.1.2.4.Çıkış Kriterlerini Değerlendirme ve Raporlama

Çıkış kriterlerini değerlendirme ve raporlama; test koşumu tamamlandıktan sonra test sonuçlarının tespit edildiği, raporlandığı ve paydaşlarca değerlendirildiği aşamadır. Test

koşumu sonucunda başlangıçta belirlenen test hedeflerinin gerçekleşme oranı saptanır. Koşum sırasında tespit edilen hatalar önceliklerine ve önemlerine göre sınıflandırılır.

Çıkış kriterleri değerlendirilirken senaryoların başarılı veya başarısız olarak netleştirilmesi; ürünün kalitesini yansıtmak açısından büyük önem taşır. Raporlama çıkış kriterlerinden oluşturulduğu için senaryoların statüleri net olarak belirlenmelidir.

Raporlamada verilerin objektif ve doğru olduğundan emin olunmalıdır. Raporlamadaki verilerden yola çıkılarak ürünün süreci hakkında paydaşlar tarafından karar verilir.

Yapılacak raporlama raporlanacak gruba göre değişiklik gösterebilir. Örneğin; üst yönetime raporlama yapılırken test koşumu sırasında tespit edilen hataları bir anlam ifade etmezken yazılım ekiplerine hazırlanan raporlamada hataların içeriği oldukça önemlidir.

2.1.2.5.Test Kapanış Aktiviteleri

Test kapanışı; test aktivitelerinin tamamlanmasına karar verildiğinde çıktıların tespit edilerek arşivlendiği aşamadır. Test çıktıları diğer benzer projelerde bakış açısı kazanmak veya gerektiğinde tekrar kullanılmak üzere arşivlenir.

Test kapanışı yapılırken bütün testlerin tamamlandığından emin olunmalıdır. Senaryolarının hepsinin koşulduğu ve hataların giderildiği veya ertelendiği teyit edilmelidir. Gerekli bilgilerin gerekli paydaşlara iletildiği kontrol edilmelidir.

Test kapanışı sırasında projeden çıkarılan dersleri belirlemek adına retrospektif toplantılar düzenlenebilir. Hangi konularda doğru veya yanlış yapıldığı teyit edilir.

2.1.3.Test Seviyeleri

Yazılım testi aktiviteleri projenin bulunduğu aşamaya göre farklı seviyelerde gerçekleştirilir. Bu seviyeler tasarlanırken yazılım geliştirme yaşam döngüsü yaşam modellerinden olan *V model* baz alınmıştır. Yazılım test seviyeleri aşağıda detaylıca belirtilmiştir.

2.1.3.1.Birim Testi

Birim testi yazılımın en küçük birimlerinin sistemden bağımsız olarak hatalarının tespit edilmesi için gerçekleştirilen testtir. Birim testi yapılırken bileşen gereksinimleri, ayrıntılı tasarım ve kod baz alınarak bileşenler, programlar, veri dönüştürme programları ve veri tabanı modülleri test edilir [14].

Birim testi fonksiyonel testlerden oluşabileceği gibi fonksiyonel olmayan testlerden de oluşabilir. Örneğin; kodun kullanılabilirliği ve fonksiyonalitesi birim testin hedefleri arasında yer alabilir.

Birim testi genellikle yazılım uzmanları tarafından gerçekleştirilir. Birim testte hataların bulunması; statik testler kadar olmasa da proje maliyetini önemli oranda düşüren bir etkendir.

Birim testinin gerçekleştirilmesi için genellikle yazılım test araçları kullanılır. Çünkü birim testinde yoğun olarak koda erişim söz konusudur.

2.1.3.2.Entegrasyon Testi

Entegrasyon testi; birimlerin arasındaki etkileşimleri, ara yüzleri test eder. Yazılım ve sistem tasarımı, mimari, iş akışları ve kullanım senaryoları baz alınarak alt sistemler, veri tabanı uyarlamaları, ara yüzler ve sistem yapılandırılması test edilir [15].

Entegrasyon testleri tasarlanırken kapsamın belirli ve kısıtlı olması; hataların birimlere indirgenmesi açısından önemlidir. Aksi halde hata giderilmesine harcanan efor artabilir.

Entegrasyon testi de birim test gibi fonksiyonel veya fonksiyonel olmayan testlerden oluşabilir. Bu testler gerçekleştirilirken birimlerin özellikleri değil aralarındaki iletişimler dikkate alınır.

2.1.3.3.Sistem Testi

Sistem testi; yazılımın tamamının teste tabi tutulduğu test seviyesidir. Test plan dokümanlarında yer alan bütün senaryolar test edilir. Sistem testinde; sistem ve yazılım

gereksinimleri, kullanım senaryoları, fonksiyonel gereksinimler ve risk analiz raporları baz alınarak sistem yapılandırması test edilir [16].

Sistem testinin ve kullanıcı kabul testinin gerçekleştirildiği ortamın gerçek ortama yakın olması oldukça önemlidir. Çünkü yazılım bu seviyeden sonra kullanıcıdan onay alındığında canlı ortama alınmaktadır.

Sistem testi de fonksiyonel ve fonksiyonel olmayan senaryolardan oluşabilir. İki durumda da test senaryoları gereksinimlerin tamamını kapsamalıdır, böylece canlı ortamda hata ile karşılaşma riski en aza indirilmiş olur.

2.1.3.4.Kullanıcı Kabul Testi

Kullanıcı kabul testleri; sistemin bütün gereksinimlerini teyit etmek amacıyla gerçekleştirilir. Kullanıcı gereksinimleri, sistem gereksinimleri, kullanım senaryoları, iş süreçleri ve risk analiz raporları baz alınarak operasyonel süreçler, kullanıcı prosedürleri ve yapılandırma verisi test edilir [16].

Kullanıcı kabul testinin gerçekleştiği ortam çoğunlukla sistem testinin gerçekleştiği ortam ile aynı ortam olur. Bu durum testler açısından efor kaybını önleyeceği gibi sistemin taşınabilirliğinin test edilmesi dikkate alınmamış olur.

Kullanıcı kabul testi son kullanıcılar tarafından gerçekleştirilir. Testlere diğer paydaşlar da katılabilir. Son kullanıcının olmadığı durumlarda kullanıcı kabul testleri test uzmanları tarafından gerçekleştirilen bir test aktivitesidir.

2.1.4.Test Çeşitleri

Yazılım testinin çeşidi test yapılan hedefe veya neden göre değişir. Aşağıda test çeşitleri detaylı olarak ele alınmıştır.

2.1.4.1.Fonksiyonel Testler

Fonksiyonel testler; yazılımın yaptıkları diğer bir deyişle gerçekleştirdiği aktiviteler dikkate alınarak yapılan testlerdir. Örneğin; yazılımın yapması gereken davranışlarının test edilmesi fonksiyonel test grubuna girer.

Fonksiyonel testler gerçekleştirilirken yazılımın gerçekleştirmesi gereken faaliyetlerin yanında gerçekleşmemesi gereken faaliyetlerin de kapsama alınması hataların tespiti için oldukça önemlidir.

Fonksiyonel testler birim testten kullanıcı kabul testlerine kadar her test seviyesinde gerçekleştirilebilir.

2.1.4.2.Fonksiyonel Olmayan Testler

Fonksiyonel olmayan testler; sistemde yapılması gerekenlerin nasıl yapıldığının dikkate alınarak gerçekleştirilen testlerdir.

Fonksiyonel olmayan testler ISO 9126 kalite standartları dikkate alınarak gerçekleştirilir. Bu testler; kullanılabilirlik, güvenilirlik, verimlilik, sürdürülebilirlik ve taşınabilirlik ana başlıklarından oluşmaktadır [17].

Fonksiyonel olmayan testlerin gerçekleştirilme zamanı; projenin kritikliğine ve kullanılan yazılım geliştirme yaşam döngüsü modeline göre değişiklik gösterir. Örneğin; Agile kullanılan bir projede tek sprinte sığmayacak olan kullanılabilirlik testleri projenin sonuna bırakılabilir.

Test planlamaları gerçekleştirilirken genellikle fonksiyonel olmayan testler dikkate alınmaz. Fakat bir ürünün çok önemli fonksiyonlara sahip olsa bile düşük performansta olması durumunda bu ürünün kullanılmamasına sebep olabileceği unutulmamalıdır. Bu sebepten test planlama aktiviteleri gerçekleştirilirken bütüncül bir bakış açısıyla yaklaşılmalıdır.

2.1.4.3.Etkileri Test Etme: Tekrar Testi ve Regresyon Testi

Test koşum sırasında hata tespit edildikten sonra hata raporu ilgili ekibe atanır. Atanan bu hata ilgili ekip tarafından analiz edilir ve çözümlenir. Bu hata raporu çözümleme işlemi sonrasında hatanın düzeldiğinin teyit edilmesi için hatayı tespit eden test uzmanına geri atanır. Bu aşamada test uzmanının hatanın giderilip giderilmediğini kontrol etmek için koştuğu teste *tekrar testi* denir. Tekrar testi sonucunda hatanın giderildiği görülürse hata raporu kapatılır. Tekrar testi sonucunda hatanın giderilmediği görülmüşse hata aynı ekibe tekrar atanır ve bu döngü hata çözülene kadar devam eder.

Hata tespit edilip çözümleme işlemi gerçekleştirildikten sonra hatanın düzeldiğinin görülmesi; her zaman sistemin tümünün doğru çalışıyor olduğunu göstermez. İlgili hata düzeltilirken yazılımın başka bir birimi bu hata düzeltme işlemi sırasındaki değişiklikten etkilenmiş olabilir. İşte bu değişikliklerin olup olmadığını sistemin sabit olduğunu teyit etmek amacıyla koşulan testlere *regresyon testi* denir. Regresyon testleri genellikle bir yazılımın projesinin bütün hataları giderildikten sonra tercih edilir. Çünkü her bir hatanın çözümlenmesinden sonra regresyon testinin gerçekleştirilmesi büyük bir efor kaybına sebep olabilir.

Tekrar testleri ve regresyon testleri tüm test seviyelerinde fonksiyonel veya fonksiyonel olmayan testlerde gerçekleştirilebilir.

2.1.5.Risk

Risk; meydana gelmesi durumunda olumsuz sonuçlara neden olabilecek durumların etkisi ve olasılığıdır [18]. Yazılım sürecinde risk; ürün canlı ortama alındığında meydana gelebilecek hataları temsil eder.

Riskler çeşitli yöntemlerle tespit edilebilir. Uzman yorumları, risk taslakları, kontrol listeleri, geçmiş projelerdeki tecrübelerden yararlanma; bu yöntemler arasında en çok kullanılanlarıdır.

Risk matematiksel olarak meydana gelebilecek hatanın olasılığı ile bu hatanın meydana gelmesi durumunda sisteme etkisinin çarpımına eşittir.

Etki; meydana gelen hatanın müşteriye, kullanıcıya veya diğer paydaşlarda bıraktığı izlenimlerdir. Etkiye itibar kaybı, iş kaybı, güven azalması vb örnekler verilebilir. Etki göz önüne alınırken hatalar kullanıcı tarafından değerlendirilir [19].

Olasılık; hatanın meydana gelme ihtimalini belirtir. Olasılığa kod karmaşıklığı, ekip içi çatışma, kullanılan teknoloji, yönetim baskısı gibi örnekler verilebilir. Olasılık göz önüne alınırken hatalar teknik açıdan değerlendirilir[19].

Riskler yapısal olarak proje riskleri ve ürün riskleri olmak üzere iki başlık altında incelenir [18]:

Proje Riskleri: Yazılımın kendisi ile ilgili olmayıp yazılımın gerçekleştirildiği projede meydana gelebilecek sorunlardır. Örneğin; Personel sorunları, sözleşme sorunları, geç kalınmış geçiş planlamaları, kod kalitesinin düşük olması vb.

Ürün Riskleri: Yazılımın kendisi ile ilgili olan sorunlardır. Bu riskler belirlenirken ISO 9126 kalite karakteristikleri dikkate alınır. Örneğin; yazılımın fonksiyonalitesi, kullanılabilirliği, verimliliği vb.

Risk azaltma faaliyetleri test ile gerçekleştirilir. Riskler en aza indirilmek için üretilen yazılım detaylıca test edilir. Gereksinimlerin tüm açıklığıyla belirtildiği ve senaryoların tüm gereksinimleri kapsadığı projeler genellikle en az hata ile canlı ortama alınabilmektedir.

2.1.6. Hata Yönetimi

Test senaryoları koşulurken test adımlarında bahsedilen beklenen sonuçlar ile gerçekleşen sonuçlar arasında farklılık olması durumuna anomali denir. Anomaliler kayıt altına alınarak raporlanır ve diğer iş ürünlerinde de gerçeği yansıtmıyorsa hata olarak nitelendirilir.

Hata raporlarının anlaşılır, objektif ve tüm detayları içeren nitelikte olması gerekir. Bir hata kaydı aşağıdaki bilgileri içermelidir [20]:

- Test edilen ortam
- Test edilen senaryo
- Beklenen ve gerçekleşen sonuçlar
- Kullanılan veri
- Hatanın tespit zamanı
- Hatanın önceliği
- Hatanın önemi
- Hatanın atandığı kişi
- Hatayı tespit eden kişi

Yukarıda belirtilen maddelerden öncelik ve önem hatanın riski ile ilgili bileşenlerdir. Öncelik hatanın ne kadar kritik olduğunu belirtir ve genellikle riskin etki faktörü ile ilişkilendirilir. Önem ise hatanın meydana gelme etkisidir ve genellikle riskin olasılık faktörü ile ilişkilendirilir [20].

2.2.TEST TASARIM TEKNİKLERİ

İlk bölümde, yazılım testi sürecinin;

- Test Planlama, Gözetim ve Kontrol
- Test Analizi ve Tasarımı
- Test Uyarlama ve Koşma
- Çıkış Kriterlerini Değerlendirme ve Raporlama
- Test Kapanış Aktiviteleri

aşamalarından oluştuğu anlatılmıştı. Test planlama aşaması tamamlandıktan sonra, test koşullarının tanımlanması için test senaryolarının dayandırıldığı dokümantasyon analiz edilmektedir. ISTQB test koşulunu, bir veya daha fazla test senaryosu ile doğrulanabilen bir durum veya olay olarak tanımlamaktadır [21]. Test koşulları tanımlanırken; risklere dayanarak kullanılacak teknikleri seçmek için yaklaşım belirlenir [22]. Test koşullarından faydalanılarak belirlenen test yaklaşımıyla test senaryolarını elde etmek veya seçmek için kullanılan tekniklere *test tasarım teknikleri* adı verilmektedir.

Test tasarım teknikleri temel olarak iki başlık altında ele alınır: statik test teknikleri ve dinamik test teknikleri. Aşağıda test tasarım tekniklerinin ayrıntılı taksonomisi mevcuttur.

2.2.1.Statik Test

Kod çalıştırılmaya başlamadan önce gereksinimlerin, analiz ve tasarım dokümanlarının gözden geçirilerek veya kodun statik analiz ile hatalarını bulmak için yapılan testlere *statik test* adı verilir. Kod çalıştırıldıktan sonra tespit edilen hataların düzeltilmesi statik test sırasında tespit edilen hataların düzeltilmesinden çok daha maliyetli olacağından yazılım geliştirme yaşam döngüsünde statik test büyük önem taşımaktadır.

Statik testler test edilen iş ürününe göre gözden geçirmeler ve statik analiz olmak üzere ikiye ayrılmaktadır.

2.2.1.1.Gözden Geçirmeler

Gözden geçirme; yazılımın ortaya çıkması için tasarım dokümanı ve gereksinim dokümanı gibi iş ürünlerinin test edilmesidir. Gözden geçirme; araç desteği bulunan ancak tamamen manuel bir işlem olarak gerçekleştirilebilen bir statik test çeşididir. Gereksinimler, tasarımlar, test planı, test gereksinimleri, test senaryoları, kullanım kılavuzları ve web sayfaları gibi tüm iş ürünleri gözden geçirilebilir. ISTQB temel seviye müfredatı gözden geçirmelerde dinamik testlerde olduğundan daha kolay bulunan genel hataları

- Standartlardan sapma,
- Gereksinim hataları,
- Tasarım hataları,
- Yetersiz sürdürülebilirlik,
- Yanlış ara yüz gereksinimleri

başlıkları altında sıralamıştır [23]. Gözden geçirmeler, test eden kişi ya da gruplara göre farklılık gösterir.

a. Gayri Resmi Gözden Geçirme

Gayri resmi gözden geçirme süreci resmi olarak ilerlemez. Ancak test koşullarının dayandırıldığı dokümantasyonların test edilecek konuda tecrübeli bir test uzmanı tarafından gözden geçirilmesini içerir. Sonuçlar kayıt altına alınabilir. Bu süreçte asıl amaç; hızlı bir şekilde hataların bulunmasıdır [24].

b. Üzerinden Geçme

Dokümanların üzerinden geçme; test senaryolarının provasını yapma aşamalarını içeren bir gözden geçirme çeşididir. Üzerinden geçme toplantıları iş ürününü oluşturan kişi tarafından yönetilir (Örneğin analiz dokümanı için süreci analist yönetir). Gözden geçirenlerin isteğine bağlı olarak toplantı öncesi hazırlık yapılabilir. Sonuçlar kayıt altına alınabilir. Bu süreçte asıl amaç; öğrenme, anlama ve hataların bulunmasıdır [24].

c. Teknik Gözden Geçirme

Eğitimli moderatör tarafından yönetilen, işveren yönetimin de dahil olduğu, hazırlanmış olan iş ürününün teknik ekip tarafından incelendiği gözden geçirme türüdür. Gözden geçiren kişiler tarafından toplantı öncesi hazırlığı yapılır. Hata listesinin gereksinimlerle karşılaştırılmasıyla teknik açıdan değerlendirme yapılır. Bu süreçte amaç; tartışarak karar verme, alternatifleri değerlendirme, hataları bulma, teknik problemleri çözme ve gereksinimlere, planlara, düzenlemelere ve standartlara uyumu kontrol etmektir [24].

d. Teftiş

Teftiş; iş ürünlerinin organizasyon içerisinde olan ayrı bir ekip tarafından detaylı incelenmesini içeren en resmi gözden geçirme çeşididir. Teftiş; konusunda eğitimli moderatör tarafından yönetilir. Tanımlı roller bulunur. Aşağıdaki aşamaları içerir.

- **Planlama:** Yapılacak olan teftiş moderatör tarafından planlanır.
- **Tanışma Toplantısı:** Teftişi gerçekleştirecek ekip ve doküman sahibi ekip arasında tanışma amaçlı bir toplantı gerçekleştirilir. Amaç kapsamlı olarak anlatılır, doküman teftiş ekibine teslim edilir.
- **Bireysel Hazırlık:** Teftiş ekibindeki gözden geçiriciler tarafından doküman tamamen gözden geçirilir. Hatalar bulunur ve raporlanır.
- **Değerlendirme Toplantısı:** Teftiş ekibi tarafından raporlanan hatalar doküman sahibi ekibe sunulur. Ekibin hataları kabul veya itirazları dinlenir. İtirazlar kabul edilir veya reddedilir.
- **Hataları Düzeltme:** Düzeltilmesi gereken hatalar doküman sahibi tarafından düzeltilir.
- **Tekrar Sunum:** Hatalar düzeltildikten sonra teftiş ekibine tekrar sunum yapılır.

2.2.1.2.Statik Analiz

Statik analiz, kod çalıştırılmadan önce kodun gözden geçirilerek yazılım hatalarının bulunmasına yönelik süreçtir. Kod çalıştırılmadan önce bulunması zor olan uyarlama hatalarını bulmaya yardımcı olur. Bunun sebebi, bu tip hataların test koşulurken bulunmasının daha zor hatta imkânsız olmasıdır. Statik analiz, uygulamayı çalıştırmaya

gerek duymadan birçok mantıksal, emniyet ve güvenlik hatalarını bulmaya yardımcı olur.

Araştırmalar statik analiz çalışmalarının gözden geçirmelere oranla daha etkili olduğunu göstermektedir. Statik analiz ile daha çok atama ve denetim gibi programlama hatalarını tespit edilirken, gözden geçirmeler fonksiyonel hataları da ortaya çıkarabilmektedir. Bununla birlikte gözden geçirmelerde insan faktörüne bağlı olarak oluşan dikkat dağılması gibi durumlar; hataların gözden kaçırılmasına neden olabilmektedir. Gözden geçirmeler zaman ve maliyet açısından statik analize göre daha az tercih edilir.

Statik analizde araç kullanımı sayesinde farklı boyutlara sahip kaynak kodlardaki hatalar benzer maliyetlerle ve çok hızlı tespit edilebilmektedir. Ayrıca yazılım hakkında istenen pek çok bilgi ve analiz sonuçları; kullanılan araçlar yardımıyla metinsel ve grafiksel olarak elde edilebilmektedir.

Statik analizde genellikle hata ve güvenlik açıkları, kaynak kod metrikleri, mimari analiz ve kodlama standartları gibi incelemeler ele alınır [25].

2.2.2.Dinamik Test

Kodun derlenip çalıştırılmaya başlanmasından sonra gerçekleştirilen testlerin tümüne *dinamik test* adı verilir. Kodlama çalışmalarının bitmesine yakın bir dönemde başlar; tespit edilen tüm hatalar çözülüp ve testin kapanışı kriterleri sağlanana kadar devam eder. Test edilecek yazılımın türüne göre, dinamik olarak uygulanacak test tasarım teknikleri ve bu tekniklerin uygulanma yöntemleri farklılık gösterebilir.

Dinamik testler temel olarak üç ana başlık altında incelenir.

- Gereksinim Bazlı Test Tasarım Teknikleri
- Yapı Bazlı Test Tasarım Teknikleri
- Deneyim Bazlı Test Tasarım Teknikleri

Aşağıda dinamik test tasarım tekniklerinin ayrıntılı açıklamaları bulunmaktadır.

2.2.2.1.Gereksinim Bazlı Test Tasarım Teknikleri

Gereksinim bazlı teknikler; kara kutu test teknikleri olarak da bilinir, iç çalışma mantığının dikkate alınmadığı ve fonksiyonel veya fonksiyonel olmayan gereksinimlerin baz alınarak sistemin test edildiği test tasarım teknikleridir. Gereksinim bazlı tekniklerde genellikle sistem gereksinim dokümanlarından faydalanılarak test senaryoları oluşturulur. Gereksinim bazlı çok sayıda test tekniği bulunmaktadır. Aşağıda en çok kullanılanlara ayrıntılı olarak yer verilmiştir.

a. Denklik Paylarına Ayırma

Denklik paylarına ayırma; aynı davranışı gösteren girdi veya çıktıların gruplandırılarak test edildiği bir test tasarım tekniğidir. Oluşturulan grupların her birinden temsili verilerin seçilip test edilmesiyle test veri sayısı azaltılmış olur. Aynı grupta olan girdilerin aynı beklenen sonucu vermesi beklenir.

Denklik payları hem geçerli veriler hem de geçersiz veriler için oluşturulabilir. Testler tüm geçerli ve geçersiz payları kapsayacak şekilde tasarlanabilir. Denklik paylarına ayırma tüm test seviyelerinde uygulanabilir. Tek boyutta ve iki boyutta ele alınmaktadır.

Denklik paylarına ayırma yöntemi tek boyutta ele alınırken yazılım girdileri aynı davranışı göstermesi beklenen gruplara ayrılır, böylece aynı şekilde ele alınabilirler.

Tek boyutta denklik paylarına ayırma tekniğini; sayıların denklik paylarına ayrılması örneğini inceleyelim. Sayıların giriş yapıldığı bir alanın olduğunu ve bu alana yalnızca rakamların girilebilmesi gereksiniminin olduğunu varsayalım. Dolayısıyla bu örnekte geçerli sınıf (0,9) olarak kabul edilirken diğer aralıklardaki tüm sayılar geçersiz olarak kabul edilir.



Şekil 2.2: Denklik Paylarına Ayırma**.**

Bu alana (0,9) aralığında değer girildiğinde hata vermemesi, 0'dan küçük ve 9'dan büyük değerlerin girilmesi halinde ise girişin geçerli olmadığına dair uyarı vermesi beklenmelidir.

Denklik paylarına ayırma yöntemi iki boyutta ele alınırken belirlenen iki parametrenin birbirlerine göre davranışları kontrol edilir ve bu davranışlara göre test senaryoları oluşturulur. Bu test senaryoları zaman kısıtı ve test kapsamına göre farklılık gösterir. Test senaryoları oluşturulurken aşağıdaki dört durum incelenir:

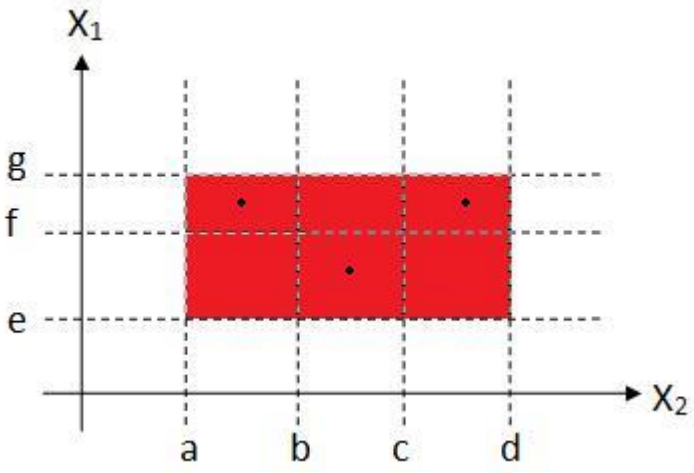
- Zayıf Normal Denklik Paylarına Ayırma
- Sağlam Normal Denklik Paylarına Ayırma
- Zayıf Kapsamlı Denklik Paylarına Ayırma
- Sağlam Kapsamlı Denklik Paylarına Ayırma

Aşağıda iki boyutta denklik paylarına ayırma yöntemlerinin ayrıntılı açıklamaları yer almaktadır.

❖ **Zayıf Normal Denklik Paylarına Ayırma**

İki değişkenli durumlar test edilirken geçerli olan aralıklardan bir ve yalnız bir test senaryosu kullanılarak yapılan denklik paylarına ayırma testleridir.

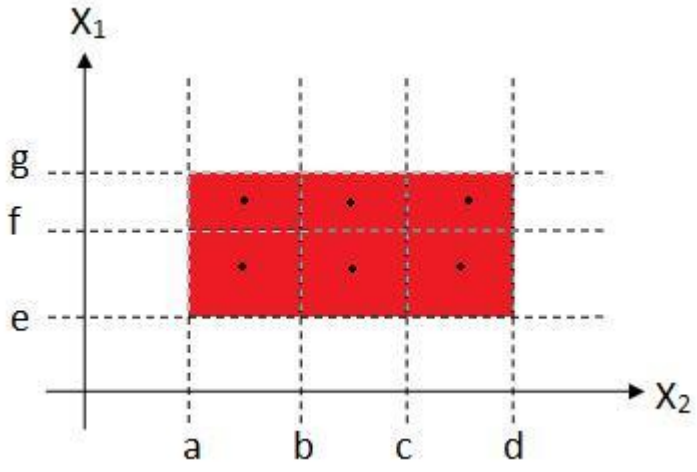
Aşağıdaki örnekte X_1 değişkeninin (e,g) aralığında X_2 değişkeninin de (a,d) aralığında geçerli olduğu görülmektedir. X_1 değişkeninin (e,f) ve (f,g) aralıklarında farklı davranışlar gösterdiği varsayılmaktadır. Benzer şekilde X_2 değişkeninin de (a,b), (b,c) ve (c,d) aralıklarında farklı davranışlar göstermektedir. Zaman kısıtı dikkate alınırsa aşağıda nokta ile belirtilen bölgelerdeki senaryolar koşularak test yapıldığında geçerli olan tüm aralıklar test edilmiş olarak kabul edilir. Fakat diğer geçerli bölgeler ve geçerli olamayan kısımlar için risk alınmış olur.



Şekil 2.3: Zayıf Normal Denklik Payı Grafiği [26].

❖ **Sağlam Normal Denklik Paylarına Ayırma**

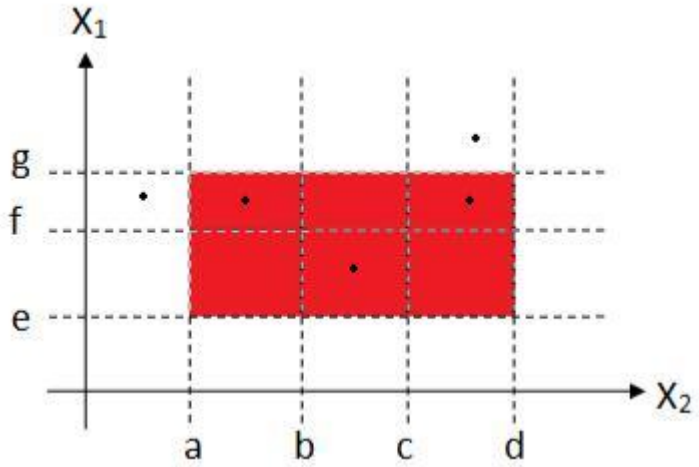
İki değişkenli testlerde geçerli olan tüm aralıklar için en az bir test senaryosu kullanılarak yapılan denklik paylarına ayırma testleridir. Aşağıdaki örnekteki gibi tüm geçerli bölgeler için bir test senaryosu koşularak geçerli olan aralıklar için risk minimuma indirilmiş olur.



Şekil 2.4: Sağlam Normal Denklik Payı Grafiği [26].

❖ **Zayıf Kapsamlı Denklik Paylarına Ayırma**

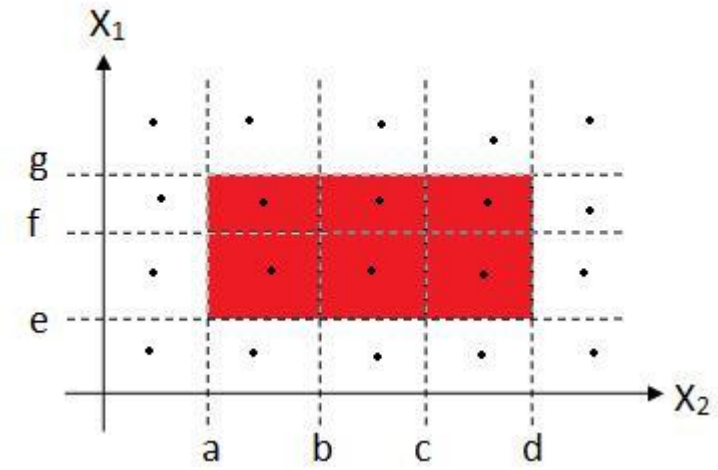
İki değişkenli testlerde geçerli ve geçersiz olan aralıklardan bir ve yalnız bir test senaryosu kullanılarak yapılan denklik paylarına ayırma testleridir. Aşağıdaki örnekte görüleceği gibi zayıf normal denklik paylarına ayırma testlerinde kullanılan test senaryolarına geçersiz olan aralıklardan en az sayıda senaryo eklenerek kapsam genişletilmiş ve geçerli olmayan aralıklardan da test senaryosu koşularak risk azaltılmıştır.



Şekil 2.5: Zayıf Kapsamlı Denklik Payı Grafiği [26].

❖ **Sağlam Kapsamlı Denklik Paylarına Ayırma**

İki değişkenli testlerde geçerli ve geçersiz olan tüm aralıklar için en az bir test senaryosu kullanılarak yapılan denklik paylarına ayırma testleridir. Bütün durumlar için test senaryosu koşulacağından kapsam bakımından risk en aza indirgenmiş olur.



Şekil 2.6: Sağlam Kapsamlı Denklik Payı Grafiği [26].

b. Sınır Değer Analizi

Sınır değer analizi; denklik paylarına ayırma tekniğinde bahsedilen denklik paylarının sınırlarının test edilmesi için kullanılan test tasarım tekniğidir. Bu teknikle sınır değerlerin doğrulaması sağlanır, doğru sınır değerinin doğru payda olduğu teyit edilir. ISTQB ileri seviye müfredatına göre; sınır değer analizi test edilen ürünün kritikliğine göre iki değerli veya üç değerli olarak gerçekleştirilir. İkili sınır değer analizinde sınırda olan değer ile sınırın hemen dışındaki değer test edilirken; üçlü sınır değerde sınırda olan, payın içinde olup sınırdan bir önceki değer olan ve sınırın dışında olup sınırdan bir sonraki değer olan değerler test edilir [27].

Sınır değer analizi tüm test seviyelerinde kullanılabilen bir tekniktir. Denklik paylarına ayırma tekniği ile beraber kullanıldığında ürüne en etkili kontrol yapılmış olur. Tek boyutta veya iki boyutta ele alınabilir.

Sınır değer analizi yöntemi tek boyutta ele alınırken sınırlardaki değerler tespit edilir. Denklik paylarına ayırma yönteminde ele alınan bir alana yalnızca rakam girilebilmesi örneğini burada da inceleyelim. Denklik paylarına ayırma yönteminde geçerli olan ve geçersiz olan değerlerin herhangi biri ile test yapılabilirken sınır değer analizinde yalnızca sınırdaki değerler ile test gerçekleştirilir. Geçerli aralık olan (0,9) aralığının sınır değerleri 0 ve 9'dur. İki değerli sınır değer analizi yapıldığında kullanılacak

değerler kümesi (-1,0,9,10) iken üç değerli sınır değer analizi yapıldığında kullanılacak değerler kümesi (-1,0,1,8,9,10) olur.



Şekil 2.7: Sınır Değer Analizi.

Sınır değer analizi tekniği iki boyutta ele alınırken denklik paylarında olduğu gibi dört farklı sınıf altında incelenir.

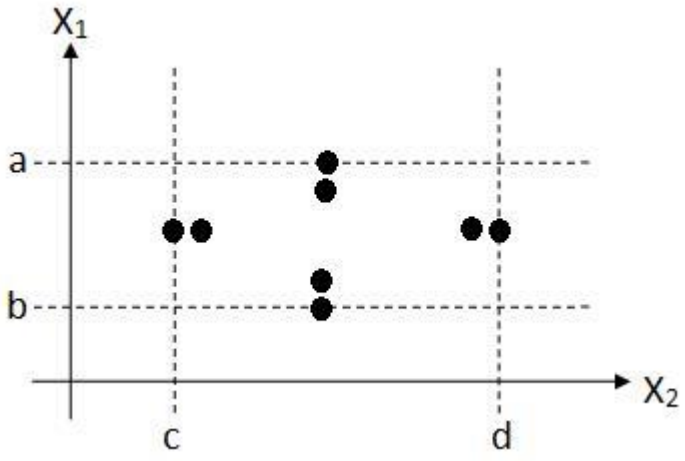
- Zayıf Normal Sınır Değer Analizi
- Güçlü Normal Sınır Değer Analizi
- Zayıf Kapsamlı Sınır Değer Analizi
- Güçlü Kapsamlı Sınır Değer Analizi

Aşağıda iki boyutta denklik paylarına ayırma yöntemlerinin ayrıntılı açıklamaları yer almaktadır.

❖ **Zayıf Normal Sınır Değer Analizi**

İki değişkenli durumları test ederken değişkenlerden yalnızca birinin sınır değerleri için oluşturulan test senaryoları kullanılarak yapılan sınır değer analizi testleridir.

Aşağıdaki örnekte X_1 değişkeninin sınırlarının a ve b, X_2 değişkeninin sınırlarının c ve d olduğu görülmektedir. Aşağıdaki grafikte nokta ile belirtilen sınırları karşılayacak test senaryoları koşularak iki değişkenin tek başlarına sınır değer analizleri test edilmiş olur. Bu yöntemde hem aynı zamanda iki değişkenin sınırları test edilmediği için hem de geçerli olmayan noktalar test edilmediği için risk alınmış olur.

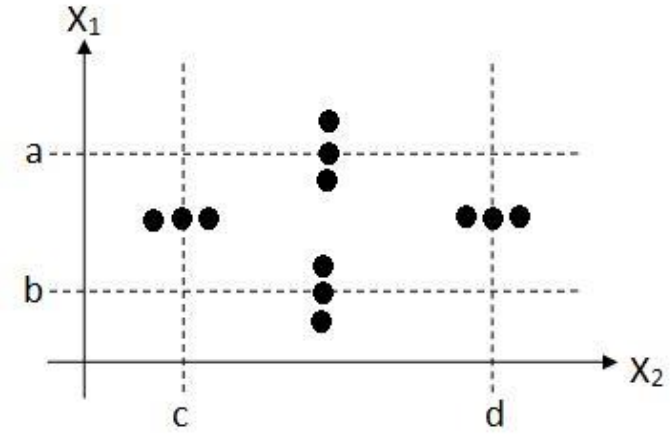


Şekil 2.8: Zayıf Normal Sınır Değer Analizi [28].

❖ **Sağlam Normal Sınır Değer Analizi**

İki değişkenli durumlarda sınırın dışındaki, sınırdaki ve sınırın içindeki değerleri ayrı ayrı ele alarak gerçekleştirilen sınır değer analizi testleridir. Zayıf normal sınır değer analizi testlerinde olduğu gibi iki değerin sınırları aynı zamanda test edilmediği için risk alınmış olur.

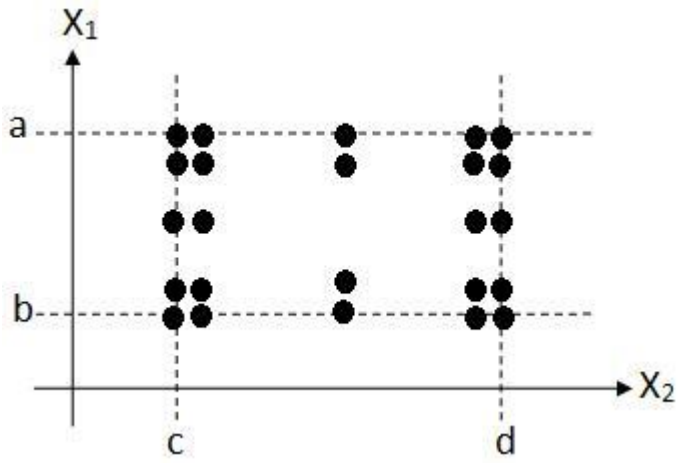
Aşağıdaki örnekte X_1 değişkeninin sınırlarının a ve b, X_2 değişkeninin sınırlarının c ve d olduğu görülmektedir. Aşağıdaki grafikte nokta ile belirtilen sınırları ve sınırların iç ve dış değerlerini karşılayacak test senaryoları koşularak iki değişkenin tek başlarına sınır değerleri test edilmiş olur. Bu yöntemde aynı zamanda iki değişkenin sınırları test edilmediği için risk alınmış olur.



Şekil 2.9: Sağlam Normal Sınır Değer Analizi [28].

❖ **Zayıf Kapsamlı Sınır Değer Analizi**

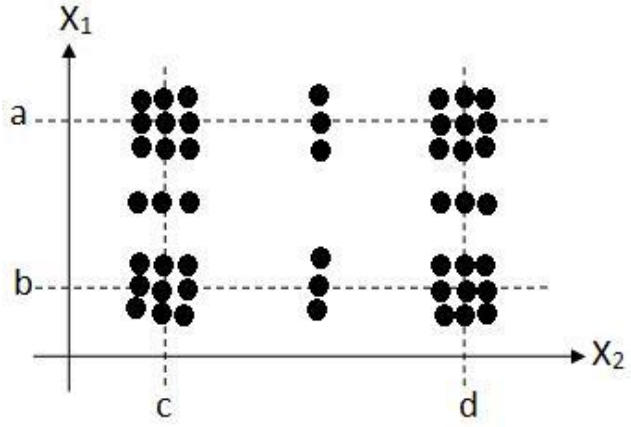
İki değişkenli durumlarda değişkenlerinin sınırlarının geçerli aralıklarda aynı anda test edilecek şekilde gerçekleştirilen sınır değer analizi testleridir. Geçerli olmayan aralıklar test kapsamında olmadığı için risk alınmış olur.



Şekil 2.10: Zayıf Kapsamlı Sınır Değer Analizi [28].

❖ **Sağlam Kapsamlı Sınır Değer Analizi**

İki değişkenli durumlarda sınırların aynı anda ve en kapsamlı şekilde test edildiği sınır değer analizi testleridir. Bütün sınır koşulları için test senaryosu koşulacağından test kapsamı bakımından risk minimuma indirilmiş olur.



Şekil 2.11: Sağlam Kapsamlı Sınır Değer Analizi [28].

c. Karar Tabloları

Karar tabloları; analiz ve gereksinim dokümanlarında belirtilen koşulların tüm kombinasyonlarının senaryolaştırılarak koşulması için kullanılan bir test tasarım tekniğidir. Koşulların kombinasyonlarının tümünün teyidini sağlaması nedeniyle ürünün kapsamlı bir şekilde test edilmesi için kullanılan en etkili yöntemlerden birisidir. Öncelikle koşullar tablonun satır kısımlarına belirtilir. Sütunların her birine de bir senaryo numarası belirtilir. Senaryo sayısı; koşul sayısı n olmak üzere 2^n olarak hesaplanır. Her bir senaryo için koşullar boolean olarak (1 ve 0) doldurulur. Sütunların sonuna çıktı değerleri belirtilir. Belirtilen test senaryoları koşularak beklenen sonuç olan çıktıların elde edildiği teyit edilir.

Zaman kısıtı kritik olan projelerde aynı çıktı değerini veren senaryolardan yalnızca birinin koşulması tercih edilebilir. Bu şekilde aynı çıktıları veren senaryolar çıkarılarak oluşturulan yeni karar tablolarına *indirgenmiş tablo* adı verilir. Karar tabloları tüm test seviyelerinde gerçekleştirilebilir.

Karar tablosu tekniğini bir örnek ile açıklayalım. Bir tatil sitesinin 25 yaş altında olan veya evli olan müşterilere indirim yaptığını varsayalım. 25 yaş altındaki müşterilere %10, evli olan müşterilere de %20 indirim yapılsın. Bu durumda tatil masrafı hesaplamalarını karar tablosu tekniği ile inceleyelim.

Örnekte iki durum bulunmaktadır:

- Birinci durum: 25 yaş altında olmak
- İkinci durum: Evli olmak

Bu iki durumdan çıkarılabilecek senaryo sayısı 2^2= 4 olur. Bu senaryolar karar tablosunda aşağıdaki gibi gösterilebilir.

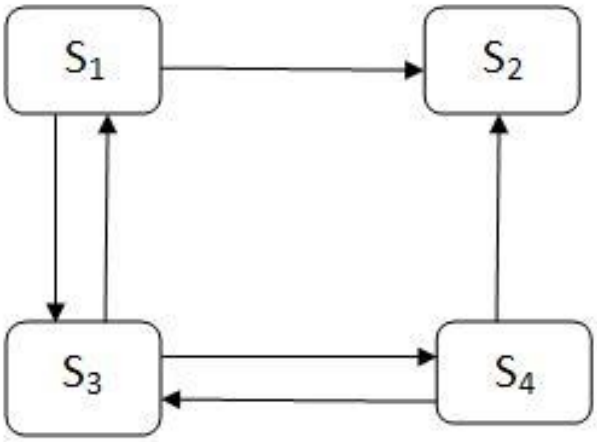
Tablo 2.1: Karar Tablosu.

	1.Senaryo	2.Senaryo	3.Senaryo	4.Senaryo
25 Yaş Altı Olmak	0	1	0	1
Evli Olmak	0	0	1	1
Sonuç (İndirim Miktarı)	0%	10%	20%	30%

d. Durum Geçişi Diyagramları

Durum geçişi diyagramları, girdilerin ve durumların arasındaki ilişkilerin test edilmesi için kullanılan bir test tasarım tekniğidir. Yazılımın tanımlanan durumlarına giriş ve çıkış kontrolleri gerçekleştirilir ve böylece geçersiz geçişler tespit edilebilir.

Durum geçişi diyagramı farklı durum seviyelerinde test edilebilir. Her geçişin bağımsız olarak test edildiği tekli geçiş kontrollerine *0-anahtarı* olarak adlandıırılırken; birbirini takip eden iki geçişin test edilmesi *1-anahtarı* olarak adlandırılır [29]. Aşağıdaki örnekte S_1, S_2, S_3 ve S_4 fonksiyonları arasındaki geçişler oklar ile gösterilmektedir.



Şekil 2.12: Durum Geçiş Diyagramı.

Diyagramdan 6 adet geçerli tek geçiş (0- anahtarı) bulunduğu anlaşılabilir.

- $S_1 \rightarrow S_2$ (S_1 'den S_2'ye)
- $S_1 \rightarrow S_3$ (S_1'den S_3'e)
- $S_3 \rightarrow S_1$ (S_3'ten S_1'e)
- $S_3 \rightarrow S_4$ (S_3'ten S_4'e)
- $S_4 \rightarrow S_2$ (S_4'ten S_2'ye)
- $S_4 \rightarrow S_3$ (S_4'ten S_3'e)

Bu geçişler dışında geçerli bir belirtilmediği için geçerli senaryolar bu geçişlerden sağlanabilir. Diğer senaryolar da geçişin sağlanmadığını kontrol etmek için kullanılabilir.

Aynı diyagramdan 8 adet ikili geçiş (1-anahtarı) bulunduğu anlaşılabilir.

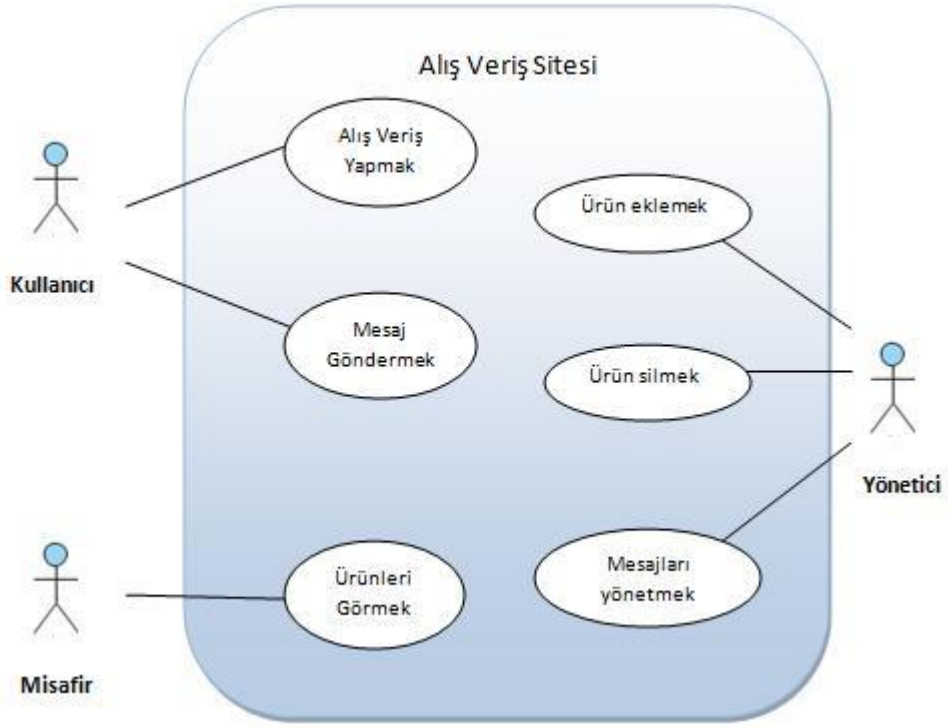
- $S_1 \rightarrow S_3 \rightarrow S_1$ (S_1 'den S_3'e, S_3'ten S_1'e)
- $S_1 \rightarrow S_3 \rightarrow S_4$ (S_1 'den S_3'e, S_3'ten S_4'e)
- $S_3 \rightarrow S_1 \rightarrow S_3$ (S_3'ten S_1'e, S_1'den S_3'e)
- $S_3 \rightarrow S_4 \rightarrow S_3$ (S_3'ten S_4'e, S_4'ten S_3'e)
- $S_3 \rightarrow S_1 \rightarrow S_2$ (S_3'ten S_1'e, S_1'den S_2'ye)
- $S_3 \rightarrow S_4 \rightarrow S_2$ (S_3'ten S_4'e, S_4'ten S_2'ye)
- $S_4 \rightarrow S_3 \rightarrow S_4$ (S_4'ten S_3'e, S_3'ten S_4'e)
- $S_4 \rightarrow S_3 \rightarrow S_1$ (S_4'ten S_3'e, S_3'ten S_1'e)

İkili geçiş senaryoları koşularak sistemin çoklu fonksiyonlarda nasıl davrandığı tespit edilebilir. Bu durum kodun canlı ortama alınmadan önce riskinin azaltılmasında oldukça etkilidir.

e. Kullanım Senaryoları

Kullanım senaryoları, sistem kullanımının simülasyonunu gerçekleştiren bir test tasarım tekniğidir. Kullanıcı ve sistem arasındaki olaylar şematik olarak ifade edilir. Bu şemalardan test senaryoları oluşturulur. Bir kullanım senaryosundan tek ya da daha fazla test senaryosu oluşturulabilir. Test senaryolarının en kolay oluşturulabildiği test tasarım tekniğidir. Bir kullanım senaryosunun test senaryolarını destekleyebilmesi için önkoşulların ve etkileşimlerin çok net bir şekilde ifade edilmiş olması gerekir. Bir kullanım senaryosunda ana senaryonun alternatif senaryoları veya hatalı olan senaryolar bulunabilir.

Aşağıda bir alışveriş sitesindeki 3 aktörün (yönetici, kullanıcı ve misafir) üzerinden ana kullanım test senaryo örneği verilmiştir. Kullanıcı için temel olan iki test senaryonun sitede alışveriş yapabilme ve mesaj gönderebilme senaryoları olduğu şekilden anlaşılmaktadır.



Şekil 2.13: Kullanım Senaryosu.

2.2.2.2.Yapı Bazlı Test Tasarım Teknikleri

Yapı bazlı teknikler, yazılım uygulamalarında kodun iç yapılarını ve alt yapılarını incelemek için uygulanan test tasarım teknikleridir. Beyaz kutu test teknikleri olarak da bilinir.

Yapı bazlı teknikler, farklı seviyelerde kodun iç çalışma mantığını farklı şekillerde irdeler:

- Bileşen seviyesi: Kod bileşen yapısı
- Entegrasyon seviyesi: Bileşenlerin arasındaki ilişkiler
- Sistem seviyesi: Menü yapısı, iş akışı

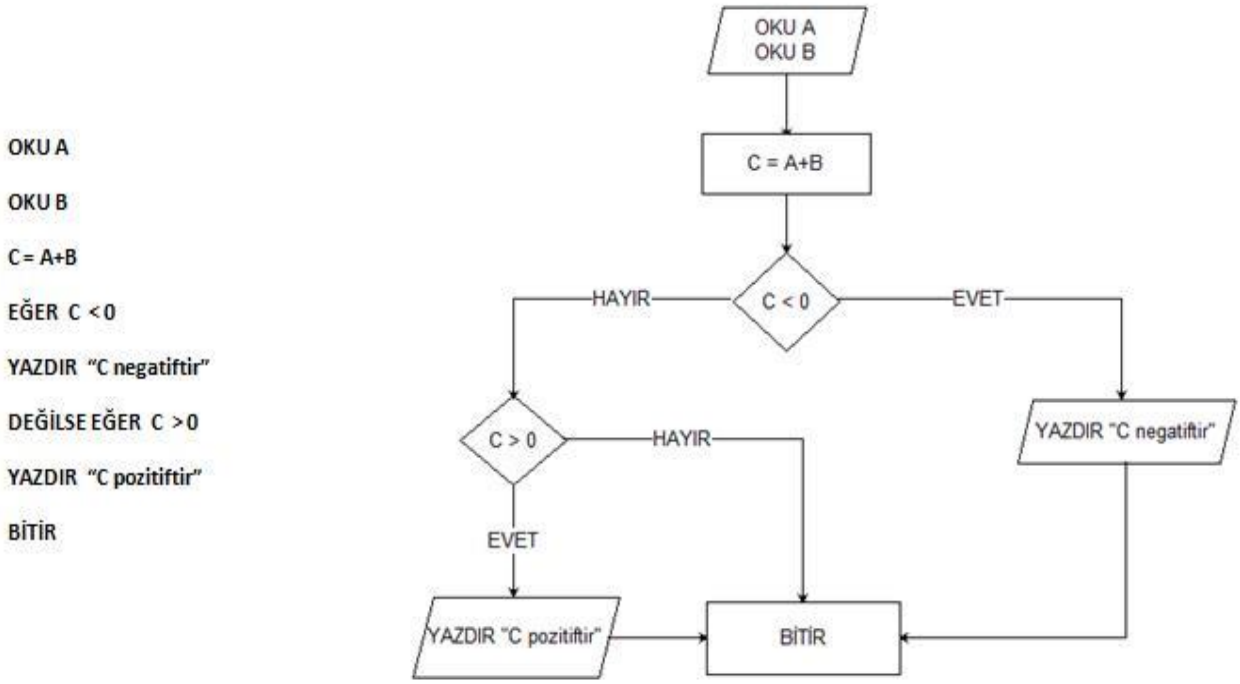
- Komut Testi
- Karar Testi

a. Komut Testi

Komut testi; kodun yapısında bulunan komutların test edildiği test tasarım tekniğidir. İlgili komutlar uygulandığında doğru çıktıların üretildiği teyit edilir.

Komut testinde kapsam komut testinin en önemli parametresi olarak kabul edilir. Senaryolarla kapsanan komut sayısının tüm komut sayısına oranı komut kapsam yüzdesini verir.

Aşağıda temel komutlarla yazılmış bir koda ait akış diyagramı bulunmaktadır. Bu koddaki komut testlerinin tamamının kontrol edilmesi için 2 senaryo koşulması yeterlidir. Bunlardan biri "C negatiftir" yazdırma komutunun uygulandığı C değişkeninin sıfırdan küçük olma durumudur. Diğeri ise "C pozitiftir" yazdırma komutunun uygulandığı C değişkeninin sıfırdan büyük olma durumudur. C değişkeninin sıfır olduğu durumda herhangi bir çıktı bulunmadığından bu durum komut testi kapsamına alınamaz.



Şekil 2.14: Komut ve Karar Testi.

b. Karar Testi

Karar testi; kodun yapısında bulunan karar yapılarının test edildiği test tasarım tekniğidir. Karar yapısının her dalı test edileceği için karar yapıları kodun tamamını test etme eğilimindedir.

Komut testinde olduğu gibi bu teknikte de kapsam en önemli parametredir. Karar kapsamı oranı, senaryolarda oluşturulmuş olan karar çıktılarının tüm karar çıktılarına oranıdır.

Karar testi gerçekleştirilirken komut testinde oluşturulan senaryolar da test edileceğinden karar testi komut testini kapsar.

Şekil 14 örneğinde karar kapsamının sağlanması için karar yapısının hem doğru hem de yanlış olarak test edilmesi gerekmektedir. Çünkü karar testinde karar yapısında bulunan bütün durumlar karar testinin kapsamındadır. Bu sebeple Şekil 14 örneğinde 3 test senaryosu koşulması karar kapsamının sağlanması için yeterlidir. Bu senaryolar;

- C değişkeninin sıfırdan küçük olma durumu,
- C değişkeninin sıfırdan büyük olma durumu,
- C değişkeninin sıfır olma durumu olarak sıralanabilir.

2.2.2.3.Deneyim Bazlı Test Tasarım Teknikleri

Deneyim bazlı test teknikleri; test edilecek konuda tecrübe ve becerilere dayandırılarak gerçekleştirilen test tasarım teknikleridir. Gereksinim bazlı ve yapı bazlı tasarım tekniklerinde tespit edilemeyen hatalar test eden kişinin tecrübesine bağlı olarak tespit edilebilir. Bu teknikler uzmanın deneyimine bağlı olduğu için yapılan testin etkisi de öznel olarak değişiklik göstermektedir.

Analiz ve gereksinim dokümanlarının olmadığı veya yetersiz olduğu durumlarda deneyim bazlı tasarım teknikleri test için oldukça önem kazanır. Deneyim bazlı test tekniklerinde testler koşulmadan önce senaryo hazırlanmaz yani test döngüsünde test tasarım ve test uyarlama eforu bulunmaz. Testlerin koşumu esas alınır. Senaryoların koşulması ve çıkış kriterlerinin değerlendirilmesi eşzamanlı olarak yapılır.

Deneyim bazlı test teknikleri dört sınıf altında incelenir:

- Keşif Testi
- Hata Tahmini
- Kontrol Listeleri
- Ataklar

a. Keşif Testi

Keşif testi; belirlenmiş zaman aralıklarında test tasarımının, test koşumunun ve sistemi analiz etmenin aynı anda gerçekleştiği bir test tasarım tekniğidir. Zamanın ve gereksinimlerin kısıtlı olduğu durumlarda sıklıkla başvurulduğu gibi diğer tekniklerin tamamlayıcısı olarak da kullanılabilir.

Keşif testinde önce test koşulur bunun sonrasında test senaryoları belirlenir. Herhangi bir ön hazırlık gerektirmeyen bir tekniktir. Öğrenmeye ve sistemi tanımaya dayalı bir teknik olduğundan senaryoların tekrarlanması her zaman mümkün olmayabilir. Genellikle en deneyimli uzmanlar tarafından zaman kısıtının olduğu durumlarda gerçekleştirilir.

Keşif testinin en önemli noktası koşulan senaryoların tekrarlanabilirliğini sağlamaktır. Çünkü kaydı tutulmayan senaryolarda hata tespit edilmesi durumunda ilgili senaryonun tekrarlanması zor olabilir.

b. Hata Tahmini

Hata tahmini; sistem üzerinde meydana gelebilecek hataların tespit edilmesine dayanan bir test tasarım tekniğidir. Testi gerçekleştiren kişinin tecrübesi, test edilen bileşen ya da sistemde hangi hataların olabileceğinin tahmin edilmesinde kullanılır.

Hata tahmini tekniğinde yazılımın yapısına bakılmadan genellikle önyüzde hatalar bulunmaya çalışılır. Tecrübeye bağlı olarak hata bulma oranı artar.

Hata tahmini diğer test tasarım tekniklerine yardımcı bir teknik olarak kullanılabileceği gibi risk analizi sırasında etkili bir yöntem olarak uygulanabilir.

c. Kontrol Listeleri

Kontrol listesi; gereksinimlerin doğrulanması için kural ve kriterlerden oluşan genel bir listenin kullanıldığı test tasarım tekniğidir. Test sırasında dikkat edilmesi gereken unsurlar genel bir liste halinde listelenir. Bu liste üzerinden test koşumu gerçekleştirilir. Test senaryosuna gerek duyulmaz.

Kontrol listeleri; standartlara, deneyime ve göz önünde bulundurulan diğer unsurlara dayanarak hazırlanır. Hata tahmini tekniğinin dokümante edilmiş versiyonu olarak düşünülebilir.

d. Ataklar

Atak; sistemin olumlu anlamda güvenlik açıklarını tespit etmek amacıyla uygulanan test tasarım tekniğidir. Bir ürünün kalitesinin, özellikle güvenilirliğinin; hataların oluşmasına zorlanarak doğrudan veya dolaylı bir şekilde test edilmesidir. Yazılıma simüle edilmiş siber saldırılar gibi penetrasyon testleri uygulanarak hataların bulunması hedeflenir.

3. MALZEME VE YÖNTEM

3.1.TEST TASARIM TEKNİKLERİ KULLANIM SIKLIĞI ANALİZİ

Yazılım testinin yazılım geliştirme yaşam döngüsündeki öneminden ve içeriğinden daha önceki bölümlerde bahsedilmişti. Geleceğe yön veren yazılım sektöründe kalitenin ve maliyetin optimum derecede sağlanma ihtiyacı; yazılım testinin önemini daha da artırmıştır. Bu durum yazılım testi alanında yoğun bir araştırma ve analiz ihtiyacını doğurmuştur. Yazılım testi uygulamaları değerlendirmesinde yazılım testi tasarım tekniklerinin kullanım sıklığı önemli parametrelerden biridir. Sektörde sıklıkla kullanılan tasarım tekniklerinin araştırılması; yazılım testi analizinde önemli çıktılar vermektedir. Bu yüzden yazılım testi tasarım tekniklerinin detaylıca araştırılması ve matematiksel modellenmesi gerekmektedir. Bu bölüm; Türkiye'deki yazılım testi alanında çalışan personelleri daha iyi analiz etmek ve kullanılan tasarım tekniklerinin kullanım sıklığını belirlemek için hazırlanan anketin çözümlemesini içermektedir.

3.1.1.Anket Soruları Analizi

Anket hazırlanırken, Türkiye'deki yazılım testi çevresini daha iyi analiz etme amacına ulaşmayı hedefleyen çıktıların saptanması için sorular aşağıdaki başlıklar bazında hazırlanmıştır:

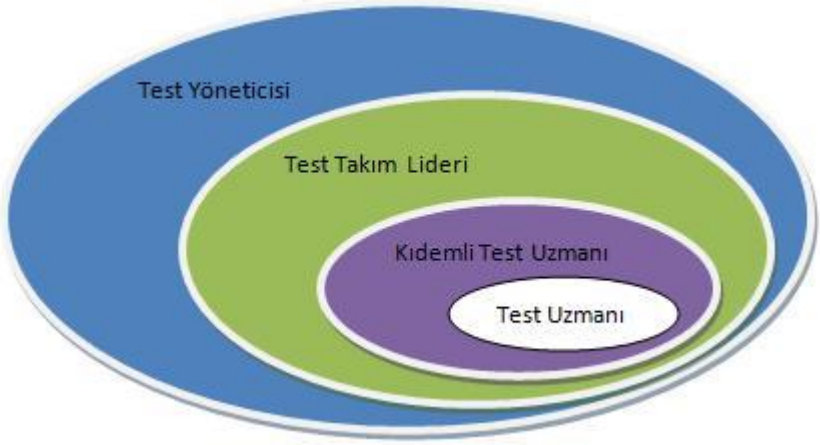
- **Katılımcı Profili:** Ankete katılan uzmanın mesleki bilgilerini, iş tecrübesini akademik derecesini öğrenmek üzere sorulan sorulardan oluşur.
- **Sektör Profili:** Ankete katılan uzmanın çalıştığı firmanın hedef sektörünü ve test ekibinin hacmini öğrenmek üzere sorulan sorulardan oluşur.
- **Proje Profili:** Ankete katılan uzmanın çalıştığı firmanın geliştirdiği yazılım türünü ve metodoloji bilgilerini öğrenmek üzere sorulan sorulardan oluşur.
- **Proje Değerlendirmeleri:** Ankete katılan uzmanın çalıştığı projelerden yola çıkarak başarı ve başarısızlık kriterlerini öğrenmek üzere sorulan sorulardan oluşur.
- **Test Tasarım Teknikleri Analizi:** Ankete katılan uzmanın test tasarım teknikleri arasındaki kullanım sıklığını analiz etmek üzere sorulan sorulardan oluşur.

Yukarıda da belirtildiği gibi ilk dört soru grubunda (katılımcı profili, sektör profili, proje profili, proje değerlendirmeleri) ankete katılan uzmanı ve bu uzmanın tecrübelerini ve bakış açısını anlamaya yönelik sorulardan oluşmaktadır. Son kısım olan test tasarım teknikleri analizine ait sorularda ankete katılan uzmanın test tasarım tekniklerini kullanma sıklığını anlamaya yönelik derecelendirme sorularından oluşmaktadır.

3.1.1.1.Katılımcı Profili

1) Mesleki Pozisyonunuz : Türkiye'de firmalar ve iş ilanları göz önüne alınarak yazılım testi alanında çalışan kişilerin aşağıdaki unvanlarda çalıştığı tespit edilmiştir.

- **Test Uzmanı:** Yazılım testi alanında çalışmaya yeni başlamış olan kişiler için kullanılan unvandır.
- **Kıdemli Test Uzmanı:** Yazılım testi alanında tecrübesi olan ve bu alanda kendini geliştirmiş olan kişiler için kullanılan unvandır.
- **Test Takım Lideri:** Yazılım testi alanında kıdemli olup aynı zamanda bir takımın sorumluluğunu almış olan kişiler için kullanılan unvandır.
- **Test Yöneticisi:** Bir firmanın bütün test organizasyonunun yönetimini üstlenmiş olan kişiler için kullanılan unvandır.



Şekil 3.1: Yazılım Testi Çalışanlarının Deneyim Kümesi.

Bu sorudan yola çıkarak katılımcıların mesleki pozisyon bilgilerinin oran olarak yoğunluğunun saptanması hedeflenmiştir.

2) İş Tecrübeniz: Bu sorudan yola çıkarak katılımcıların iş tecrübelerinin oran olarak yoğunluğunun saptanması hedeflenmiştir.

3) Akademik Dereceniz: Türkiye'de yazılım testi alanında iş ilanları baz alınıp aşağıdaki akademik derecelere sahip oldukları düşünülerek bu seçenekler sunulmuştur.

- Doktora
- Yüksek Lisans
- Lisans
- Önlisans
- Lise ve altı

Bu sorgulamada amaç; katılımcıların akademik derecelerinin oran olarak yoğunluğunun saptanmasıdır.

4) Akademik Derece Aldığınız Alanlar: Türkiye'de yazılım testi alanında iş ilanları baz alınıp aşağıdaki akademik alanlardan mezun oldukları düşünülerek bu seçenekler sunulmuştur.

- Bilgisayar/Bilişim Sistemleri Mühendisliği
- Yazılım Mühendisliği
- Elektrik/Elektronik Mühendisliği
- Matematik Mühendisliği
- Endüstri Mühendisliği
- Diğer(Lütfen belirtiniz.)

Bu sorgulamada amaç; katılımcıların ihtisas alanlarının oran olarak yoğunluğunun saptanmasıdır.

3.1.1.2.Sektör Profili

Bu bölümde sorulan sorular ile uzmanın çalıştığı alan ve test organizasyon büyüklüğünün analizi mümkün hale gelir.

1) Çalıştığınız firma tarafından geliştirilen ürünlerin hedef sektörü nedir?: Test alanında iş ilanları baz alındığında en çok aşağıdaki hedef sektörlerde test ihtiyacı olduğu görülmüştür ve uzmanlara bu seçenekler sunulmuştur.

- Bankacılık/Finans
- Savunma Sanayi
- Telekom
- Sigorta
- İşletme
- Otomotiv
- Diğer(Lütfen belirtiniz.)

Bu sorgulamada amaç; katılımcıların hizmet verdiği sektörlerin oran olarak yoğunluğunun saptanmasıdır.

2) Çalıştığınız firmada test ekibi personel sayısı kaçtır?: Türkiye'de test konusunda ilerleme kaydetmiş olan firmalar göz önüne alınarak personel sayısı aşağıdaki sınıflara ayrılmıştır.

- 1-10
- 11-30
- 31-50
- 50'den fazla

Bu sorgulamada amaç; firmanın teste yaptığı yatırımın belirlenmesidir. Ankette teste verilen önemin belirlenmesi için sorulan serbest sorulardan birisidir.

3.1.1.3.Proje Profili

Bu bölümde sorulan sorular ile uzmanın testini gerçekleştirdiği yazılım projelerinin özelliklerini ve kullanılan yazılım metodolojisini anlamak ve test tasarım teknikleri ile arasındaki ilişkiyi saptamak mümkün hale gelir.

1) Firmanızda geliştirdiğiniz yazılım paketlerini nasıl sınıflandırırsınız?: İş ilanları ve uzman görüşleri baz alınarak firmalarda aşağıdaki yazılım paketlerinin geliştirilebileceği öngörülmüştür.

- Web Uygulamaları
- Mobil Uygulamalar
- Sistem Yazılımı
- Diğer(Lütfen belirtiniz.)

2) Projeleriniz hangi metodolojide işletiliyor? : Firma yapıları göz önüne alınarak yürütülen projelerin aşağıdaki metodolojiler ile yürütüldüğü öngörülmüştür.

- Waterfall
- V Model
- Spiral Model
- Agile
- Diğer(Lütfen belirtiniz.)

3.1.1.4.Proje Değerlendirmeleri

Bu bölümde uzmana yöneltilen sorular test tasarım tekniklerinden biraz daha bağımsız, serbest olan sorulardır. Bu sorular ile test uzmanının perspektifinden projelerin başarı ve başarısızlık nedenleri tespit edilmek istenmiştir. Bu bölümdeki cevaplar ürünün kalite kontrolünü gerçekleştiren test uzmanlarının proje başarı kriterlerini en net görebilen kişiler olduğu düşünülerek sektör açısından büyük önem taşımaktadır.

1) Firmanızda projelerin başarılı olmasındaki en önemli kıstas sizce hangisi? : Projelerin başarı kriterlerinin aşağıdaki seçenekler olduğu düşünülerek uzmanlara bu seçenekler sunulmuştur.

- Gereksinimlerin açık ve net olarak belirlenmiş olması
- Kaliteli kod
- Kaliteli test süreci
- Diğer(Lütfen belirtiniz.)

2) Firmanızda projelerin başarısız olmasının temel nedeni sizce hangisi?: Projelerin başarısızlık kriterlerinin aşağıdaki seçenekler olduğu düşünülerek uzmanlara bu seçenekler sunulmuştur.

- Gereksinimlerin açık olarak belirtilmemiş olması
- Yönetim ve zaman baskısı
- Kalitesiz sdlc süreci
- Diğer(Lütfen belirtiniz.)

3.1.1.5. Test Tasarım Teknikleri Analizi

Bu bölümdeki sorular uzmanın tasarım tekniklerini önceliklendirmesini ve tekniklerin kullanılma oranını saptamak için oluşturulmuştur. Uzmanların teknikleri hatırlaması ve bilmediği teknikler hakkında fikir sahibi olması açısından soruların alt kısımlarına ikinci bölümde bahsedilen test tasarım tekniklerinin kısa tanıtım bilgileri eklenmiştir.

1) Firmanızda gerçekleştirilen test tasarım tekniklerini işaretleyiniz: İkinci bölümde bahsedilen test tasarım teknikleri seçenek olarak sunulmuştur.

- Black-box test teknikleri(denklik paylarına ayırmak sınır değer, karar tablosu gibi)
- White-box test teknikleri(durum testi, karar testi)
- Tecrübeye dayalı teknikler(hata tahmini, keşif testi gibi)

Bu sorgulamada amaç; katılımcı tarafından kullanılan test tasarım tekniklerini belirlemektir.

2) Gereksinim bazlı test tekniklerini puanlayınız: Bu soruda ikinci bölümde de bahsedilen gereksinim bazlı test teknikleri seçenek olarak sunulmuştur.

- Denklik Paylarına Ayırma
- Sınır Değer Analizi
- Karar tablosu
- Geçiş Diyagramları
- Kullanım Senaryoları

Bu soruda kullanıcıdan her bir tekniği 1 ile 10 arasında bir sayı ile puanlaması istenmiştir. Puanlamayı yaparken bu teknikleri kullanma sıklığı ve projelerle bu tekniklerin uyumluluğunun göz önünde bulundurulması gerektiği hatırlatılmıştır.

3) Yapı bazlı test tekniklerini puanlayınız.: Bu soruda ikinci bölümde de bahsedilen yapı bazlı test teknikleri seçenek olarak sunulmuştur.

- Komut Testi
- Karar Testi

Bu soruda kullanıcıdan her bir tekniği 1 ile 10 arasında bir sayı ile puanlaması istenmiştir. Puanlamayı yaparken bu teknikleri kullanma sıklığı ve projelerle bu tekniklerin uyumluluğunun göz önünde bulundurulması gerektiği hatırlatılmıştır.

4) Deneyim bazlı test tekniklerini puanlayınız. : Bu soruda ikinci bölümde de bahsedilen gereksinim bazlı test teknikleri seçenek olarak sunulmuştur.

- Keşif Testi
- Hata Tahmini
- Kontrol Listeleri
- Saldırılar

Bu soruda kullanıcıdan her bir tekniği 1 ile 10 arasında bir sayı ile puanlaması istenmiştir. Puanlamayı yaparken bu teknikleri kullanma sıklığı ve projelerle bu tekniklerin uyumluluğunun göz önünde bulundurulması gerektiği hatırlatılmıştır.

5) Firmanızda statik test ne sıklıkla yapılmaktadır? Türkiye'de statik test genellikle önem verilmeyen ve gereksiz görülen veya zaman kısıtından dolayı gerçekleştirilemeyen bir konudur. Bu soru ile firmaların ve uzmanların statik teste verdiği değerin saptanması hedeflenmiştir. Soruda aşağıdaki seçenekler sunulmuştur.

- Hiç
- Bazen
- Analiz dokümanları erken geldikçe
- Her zaman

6) Statik test tekniklerini puanlayınız: Bu soruda ikinci bölümde de bahsedilen statik test teknikleri seçenek olarak sunulmuştur.

- Gayri Resmi Gözden Geçirme

- Üzerinden Geçme
- Teknik Gözden Geçirme
- Teftiş

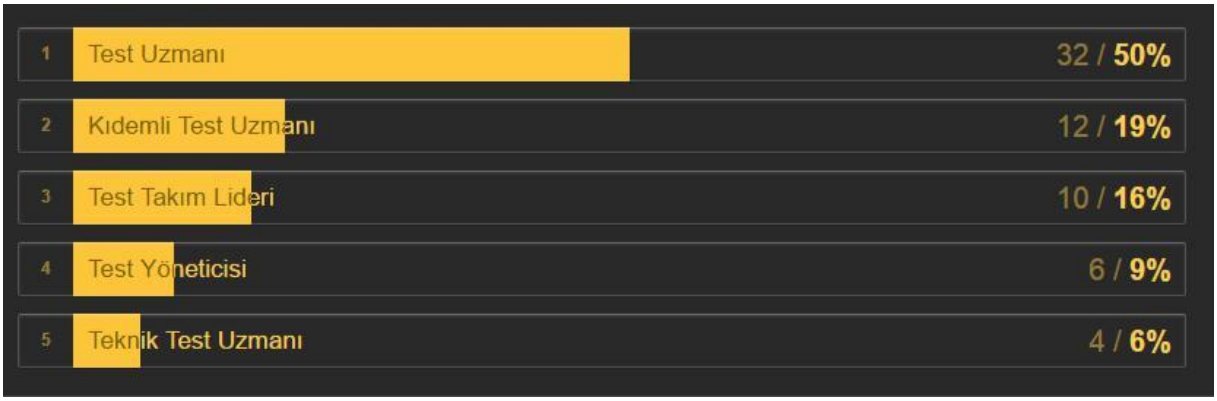
Bu soruda kullanıcıdan her bir tekniği 1 ile 10 arasında bir sayı ile puanlaması istenmiştir. Puanlamayı yaparken bu teknikleri kullanma sıklığı ve projelerle bu tekniklerin uyumluluğunun göz önünde bulundurulması gerektiği hatırlatılmıştır.

3.1.2.Anket Sonuçları Analizi

Anket 22 Nisan 2016 -1 Temmuz 2016 tarihleri arasında gerçekleştirilmiştir. Ele alınan konu yazılım testi olduğundan yazılım testinde alanında çalışan 65 uzmanın görüşlerine başvurulmuştur. Aşağıda her bir soru ile ilgili sonuç analizi yer almaktadır.

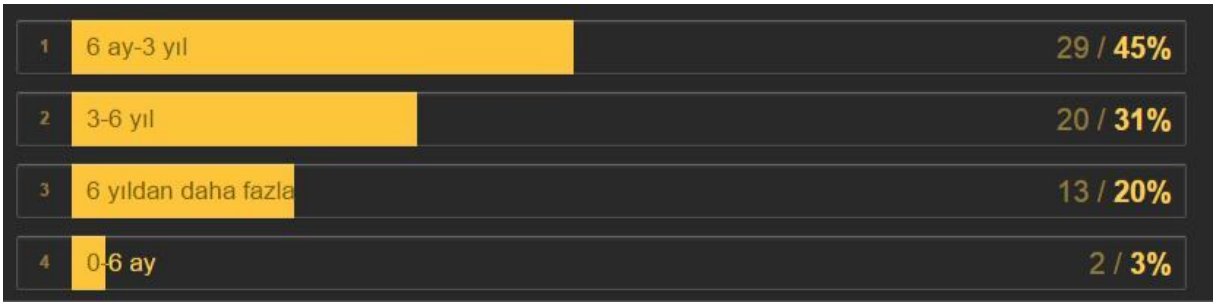
3.1.2.1.Katılımcı Profili

1) Mesleki Pozisyonunuz: Bu sorunun cevap dağılım yüzdesi aşağıdaki gibidir.



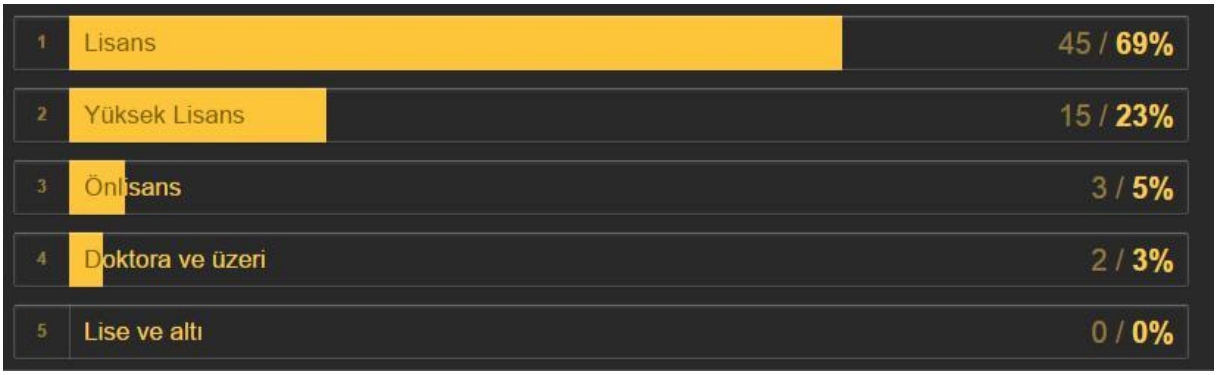
Ankete katılan büyük çoğunluğun uzman pozisyonunda olduğu rahatça anlaşılabilmektedir. Fakat diğer pozisyonlardan da yeterli sayıda uzman bulunmaktadır. Matematiksel modelleme için her pozisyon grubundan yeterli sayıda uzman katılmıştır.

2) İş Tecrübeniz: Bu sorunun cevap dağılım yüzdesi aşağıdaki gibidir.



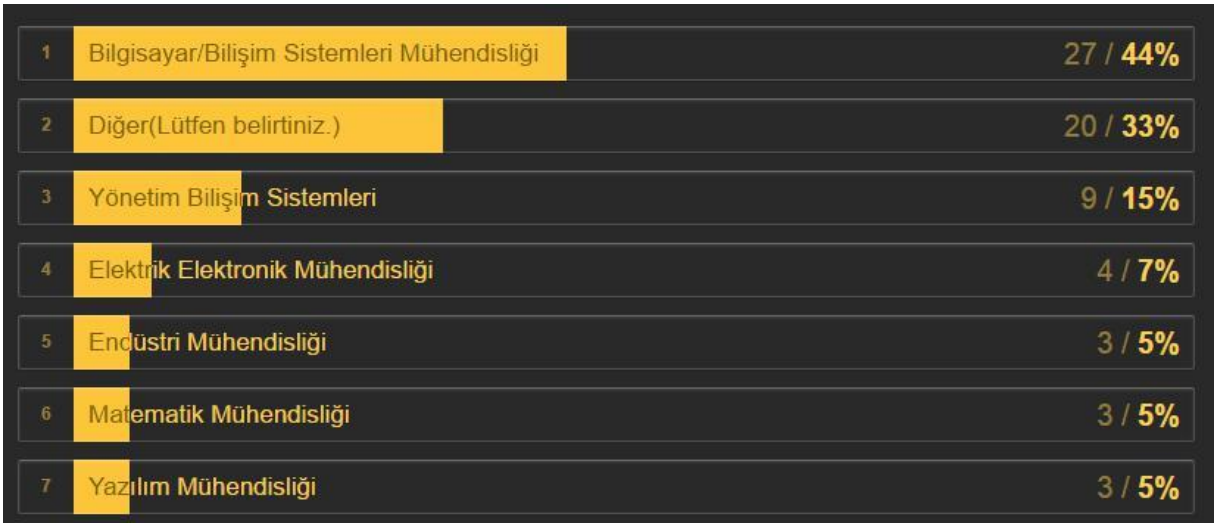
Tecrübesi olmayan adaylar dışında ankete katılan adayların tecrübe sürelerinin dengeli olduğu söylenebilir. Test konusunda henüz az bilgisi olan uzmanların sayısının az olması, anket sonuçlarının daha doğru olmasını sağlamıştır.

3) Akademik Dereceniz: Bu sorunun cevap dağılım yüzdesi aşağıdaki gibidir.



Yukarıdaki yüzdelerden test mühendisliği pozisyonu için seçilen personellerin genellikle lisans mezunu ve üstü akademik dereceye sahip olmaları gerektiği sonucu çıkarılabilir.

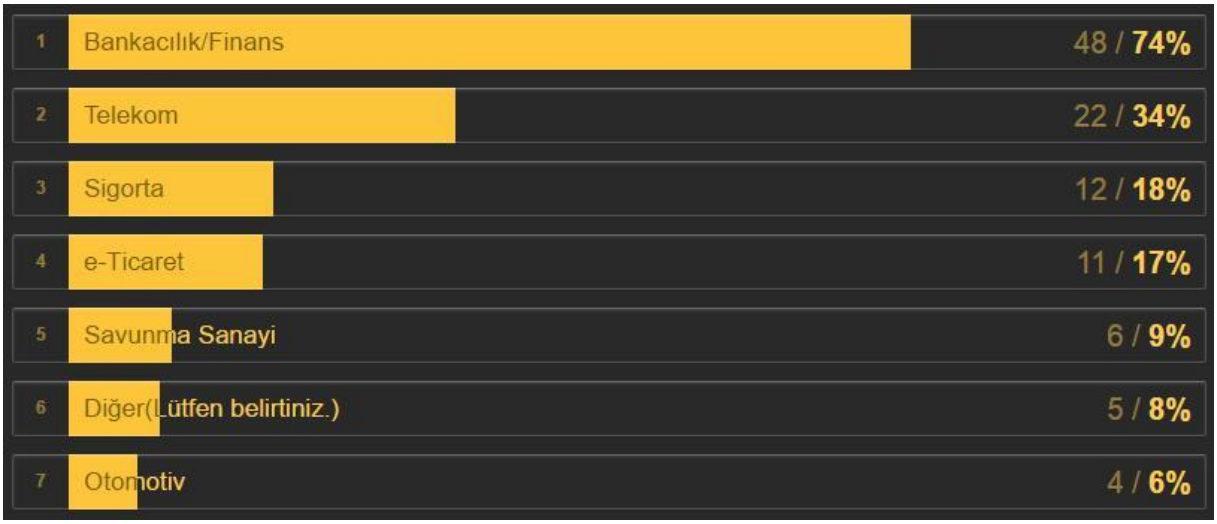
4) Akademik Derece Aldığınız Alanlar: Bu sorunun cevap dağılım yüzdesi aşağıdaki gibidir.



Yukarıdaki yüzdelerden test mühendisliği pozisyonu için en çok tercih edilen akademik alanın bilgisayar ve bilişim sistemleri olduğu açıkça görülmektedir. Diğer yandan iş ilanlarında belirtilen mühendislikler dışında bu alanda çalışan diğer akademik alan mezunlarının da sayısının oldukça fazla olduğu sonucu çıkarılabilir. 'Diğer' seçeneğini işaretleyen adayların akademik alanlarına bakıldığında büyük çoğunluğunun **diğer mühendislik türleri**, **doğa bilimleri** ve **iktisadi bilimler** mezunu oldukları görülmektedir. Bunun sebebinin yazılım testinin Türkiye'de uygulanmaya başlandığı dönemlerde nitelikli uzman bulunamamasından kaynaklandığı yorumlanabilir.

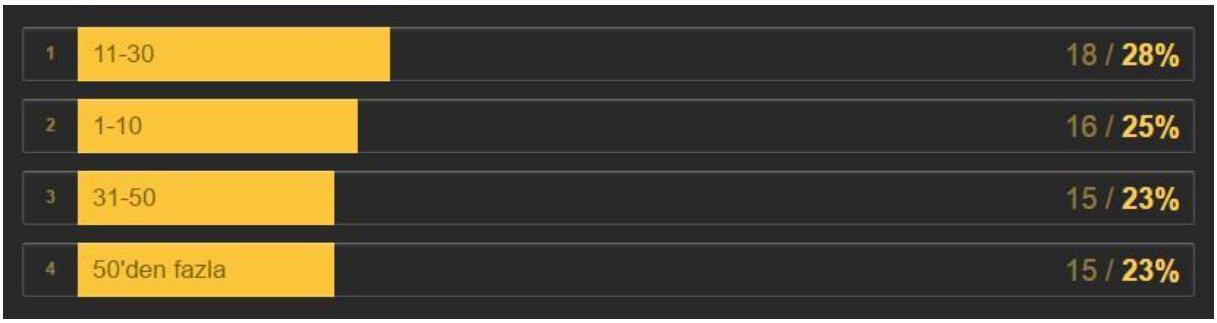
3.1.2.2.Sektör Profili

1) Çalıştığınız firma tarafından geliştirilen ürünlerin hedef sektörü nedir?: Bu sorunun cevap dağılım yüzdesi aşağıdaki gibidir.



Yüzdelere bakıldığında çok homojen bir dağılım olmadığı görülmekle beraber bankacılığın; test tasarım tekniklerinin en iyi uygulandığı iş alanlarından biri olduğu düşünüldüğünde anketlerdeki sonuçların daha doğru oranda elde edilebileceği düşünülebilir. Soruyu 'Diğer' olarak cevaplayan uzmanlar sağlık sektöründe çalıştıklarını belirtmişlerdir.

2) Çalıştığınız firmada test ekibi personel sayısı kaçtır? Bu sorunun cevap dağılım yüzdesi aşağıdaki gibidir.

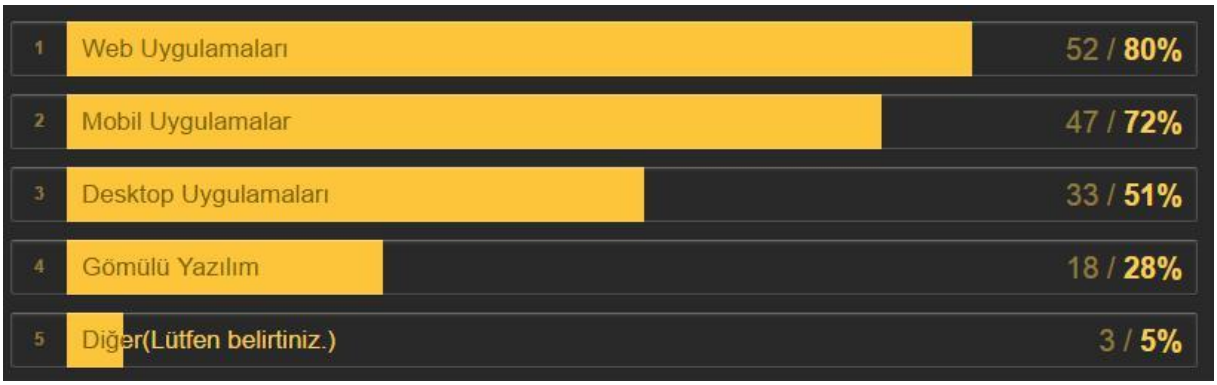


Bu tablodan Türkiye'de artık test faaliyetlerine önem verildiği ve bu konuda kayda değer yatırımlar yapıldığı sonucunu çıkarabiliriz. 10 yıl öncesine kadar yan faaliyet olarak görülen, dikkate alınmayan yazılım testinin ilerde de ihtiyaçlar doğrultusunda

daha kalabalık ekipler tarafından gerçekleştirilen bir yazılım geliştirme yaşam döngüsü aktivitesi olacağı düşünülmektedir.

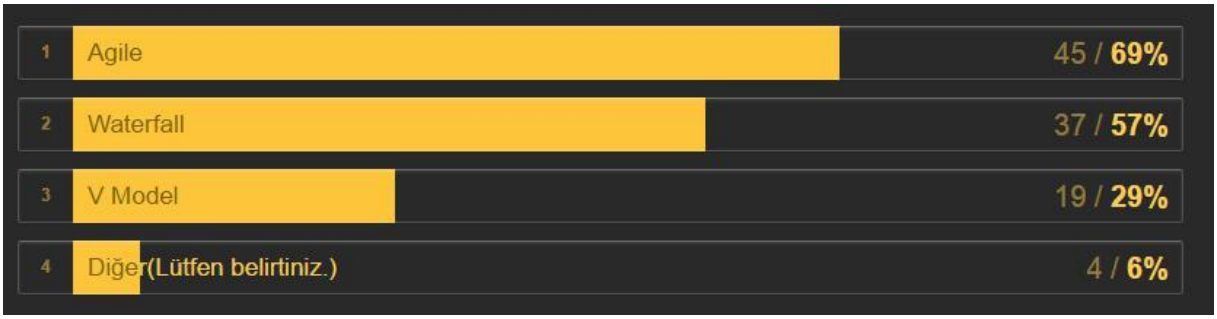
3.1.2.3.Proje Profili

1) Çalıştığınız firmada geliştirilen yazılım paketlerini nasıl sınıflandırırsınız? : Bu sorunun cevap dağılım yüzdesi aşağıdaki gibidir.



Geliştirilen ürünlerin çoğu web ve mobil uygulaması olarak hizmete sunulmaktadır. Bunların yanında özellikle bankacılık ve finans alanlarında sıkça kullanılan desktop uygulamaları ve savunma sanayisi alanında sıkça kullanılan gömülü yazılım geliştirmeleri de hizmete fazlaca sunulan yazılım paket türlerindendir. 'Diğer' seçeneğini belirten uzmanların da yine web ve gömülü yazılımı tercih ettikleri görülmüştür.

2) Çalıştığınız firmada projeleriniz hangi metodolojide işletilmektedir? : Bu sorunun cevap dağılım yüzdesi aşağıdaki gibidir.

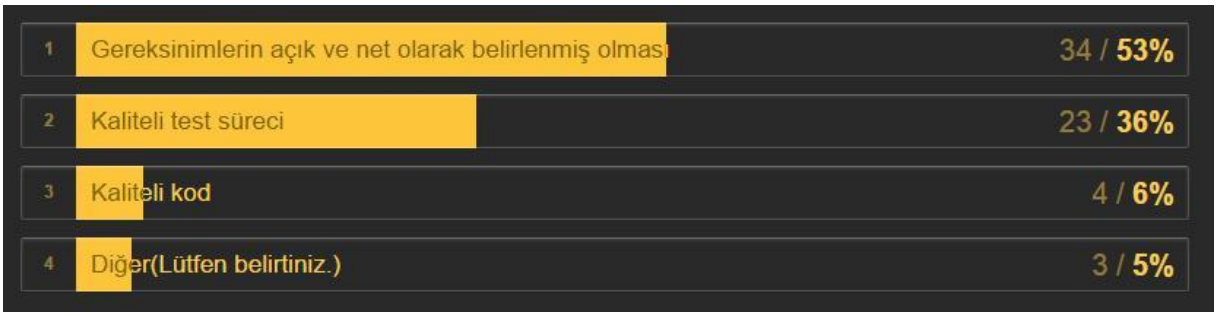


Yüzdelere bakıldığında dünya genelinde artık iyice hâkim olmaya başlamış olan Agile metodolojisinin kullanılmakta olduğu görülmektedir. Onu takip eden waterfall ve

V model modelinin daha çok bankacılık, sigortacılık ve finans gibi değişikliklere adaptasyon konusunda biraz daha yavaş yol alan alanlarda tercih edildiği görülmüştür. “Diğer” olarak belirtilen cevaplara bakıldığında firmaların kendilerine özgü oluşturmuş olduğu metodolojilerin kullanıldığı görülmüştür.

3.1.2.4.Proje Değerlendirmeleri

1) Çalıştığınız firmada projelerin başarılı olmasındaki en önemli kıstas sizce hangisi? : Test uzmanlarının bakış açısından projelerin başarı kriterlerini analiz etmek için sorulan bu sorunun cevap dağılım yüzdesi aşağıdaki gibidir:



Cevaplardan açıkça anlaşılacağı üzere test uzmanlarının büyük çoğunluğu projenin başarılı olmasının gereksinimlerin açık ve net olarak belirlenmiş olmasına bağlı olduğunu savunmaktadır. Bir diğer çoğunluk ürünün test sürecinin kaliteli olmasının başarıyı etkileyen en önemli faktör olduğunu düşünmektedir. Bunlar dışında kaliteli kodun başarının en önemli faktör olduğunu düşünen az sayıda test uzmanı vardır. ‘Diğer’ seçeneğini cevaplayan test uzmanlarının cevapları aşağıdaki gibidir:

- Projeler başarılı gerçekleştirilememektedir.
- Projenin verilen zamandan daha erken bitirilmiş olması en önemli etkendir.
- Müşteriyi memnun etmesi başarılı olması açısından en önemli etkendir: Bu cevap gereksinimlerin açık ve net karşılanmış olması ile aynı nedene dayanmaktadır. Çünkü gereksinimler müşterilerden fikir alınarak oluşturulan sistem ihtiyaçlarıdır.

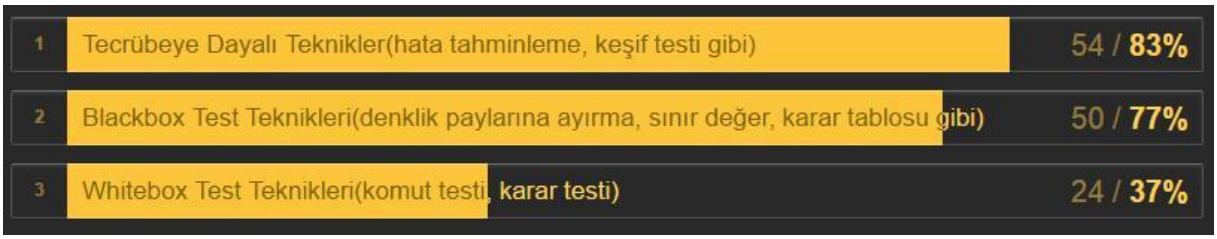
2) Çalıştığınız firmada projelerin başarısız olmasının temel nedeni sizce hangisi? : Test uzmanlarının bakış açısından projelerin başarısızlık kriterlerini analiz etmek için sorulan bu sorunun cevap dağılım yüzdesi aşağıdaki gibidir:



Uzmanların bir kısmı yönetim ve zaman baskısının projeyi başarısızlığa götürdüğünü savunmaktadır. Diğer bir kısım gereksinimlerin açık ve belirtilmemiş olmasının sürece baştan yanlış başlanmasını ve bu yanlışlığın yazılım geliştirme yaşam döngüsü boyunca devam ederek projenin başarısızlıkla sonuçlandığını düşünmektedir. Yetersiz yazılım geliştirme yaşam döngüsü sürecinin başarısızlığın temel nedeni olduğunu düşünen uzman sayısı da az sayılmayacak kadar fazladır. 'Diğer' seçeneğini cevaplayan test uzmanlarının cevapları incelendiğinde iletişimsizliğin projeyi başarısızlığa götüren temel nedeni savundukları görülmüştür.

3.1.2.5. Test Tasarım Teknikleri Analizi

1) Çalıştığınız firmada testlerinizi yaparken kullandığınız test tasarım tekniklerini işaretleyiniz: Bu sorunun cevap dağılım yüzdesi aşağıdaki gibidir.



Uzmanlara çoklu cevap verebilecekleri belirtilmiş ve büyük çoğunluğunun tecrübeye dayalı test tekniklerini tercih ettikleri görülmüştür. Onu az bir fark ile gereksinim bazlı test teknikleri takip etmektedir. Kodun iç yapısını inceleyen yapı bazlı test teknikleri de diğer iki teknik gruba göre daha az tercih edilmektedir. Bunun sebebinin kodun iç

yapısını bilen ve bu konuda eğitim almış olan test uzmanlarının sayısının az olması olduğu düşünülmektedir.

2) Gereksinim Bazlı Test Tekniklerinin Kullanılma Sıklığına Göre Puanlanması:
Bu kısımda her bir test tekniğinin kullanım sıklığı ve firmaya uygunluğu açısından puanlandırılması istenmiştir. Gereksinim bazlı test tekniklerine verilen puanların ortalaması aşağıdaki gibidir

- Denklik Paylarına Ayırma



- Sınır Değer Analizi



- Karar Tablosu



- Durum Geçiş Diyagramları



- Kullanıcı Senaryosu



Gereksinim bazlı test tekniklerini puanlayan uzmanların yazılım metodolojilerine göre dengeli olarak dağıldığı görülmüştür. Waterfall, V-model ve Agile metodolojilerinin üçünde de gereksinim bazlı test tekniklerinin yüksek oranda kullanıldığı bilgisi çıkarılabilir. Yazılım paketi sınıfına göre incelendiğinde ise masaüstü uygulamaları ve gömülü yazılımlarda gereksinim bazlı test tekniklerinin daha fazla tercih edildiği saptanmıştır.

3) Yapı Bazlı Test Tekniklerinin Kullanılma Sıklığına Göre Puanlanması: Yapı bazlı test tekniklerine verilen puanların ortalaması aşağıdaki gibidir.

- Komut Testi



- Karar Testi



Yapı bazlı test tekniklerinin çok fazla kullanılmadığı anket sonuçlarından açıkça görülebilmektedir. Yapı bazlı test tekniklerinin kullanım sıklığını yüksek puanlayan uzmanların genellikle gömülü yazılım tipindeki projelerde çalıştıkları görülmüştür. Gömülü yazılımda komut ve karar testlerinin, yazılım testinin önemli bir parçası olduğu düşünülebilir.

4) Deneyime Dayalı Test Tekniklerinin Kullanım Sıklığına Göre Puanlanması:

Deneyime dayalı test tekniklerine verilen puanların ortalaması aşağıdaki gibidir.

- Keşif Testi



- Hata Tahmini



- Kontrol Listesi

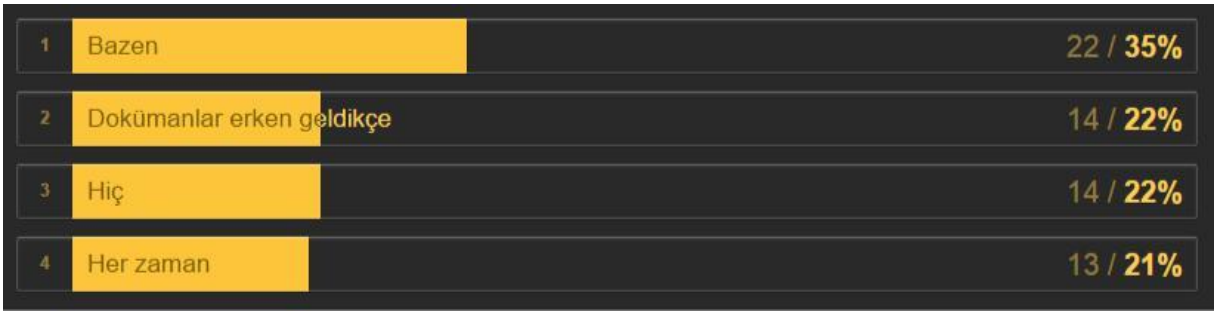


- Ataklar



Deneyime dayalı test tekniklerinin çoğunluğunun sıkça kullanıldığı puan ortalamalarından açıkça görülebilmektedir. Anket sonuçlarına göre bu tekniklerin en çok Agile metodolojisi ile çalışan uzmanlar kullanmaktadır. Agile modelde deneyime dayalı test teknikleri tamamlayıcı bir etkiye sahip olduğundan tercih edilmesi muhtemeldir. Diğer yandan bütün test teknikleri içerisinde atak türünün en az kullanılan teknik olduğu görülmüştür.

5) Çalıştığınız firmada ne sıklıkla statik test yapılmaktadır?Bu sorunun cevap dağılım yüzdesi aşağıdaki gibidir.



Anket sonuçlarından statik test alışkanlığının henüz tam anlamıyla yerleşmediği görülebilmektedir. Diğer yandan statik testi sürekli yaptıklarını belirten uzmanların çoğunlukla savunma ve sağlık sektörlerinde çalıştıkları görülmüştür. Bu sonuçtan da açıkça görülebileceği üzere savunma ve sağlık gibi güvenliği oldukça önemli olan geliştirmelerde statik test oldukça büyük bir öneme sahiptir.

6) Statik Test Tekniklerinin Kullanım Sıklığına Göre Puanlanması: Statik test tekniklerine verilen puanların ortalaması aşağıdaki gibidir.

- Gayri Resmi Gözden Geçirme



- Üzerinden Geçme



- Teknik Gözden Geçirme



- Teftiş



- Statik Analiz



Statik test tekniklerinin kullanım sıklığı ortalamalarına bakıldığında en çok kullanılan tekniğin ‘Üzerinden Geçme’ olduğu görülmüştür. Diğer yandan yapısal bazlı test tecrübesi olan uzmanların daha çok teknik gözden geçirme ve statik analiz tekniklerini tercih ettikleri görülmüştür.

3.2.TEST TASARIM TEKNİKLERİ HATA BULMA ORANLARI ANALİZİ

Bu bölümde matematiksel modelleme için belirlenen ikinci parametre olan hata bulma oranlarının analizi yer almaktadır. Test tasarım tekniklerinde uzman kişiler için hazırlanan hata tespit etme oranı çalışmasının içerik çözümlemesi gerçekleştirilecektir.

Hata bulma oranlarını analiz etmek için gereksinim bazlı test teknikleri konusunda deneyimli 8 uzmanın, yapı bazlı test teknikleri konusunda deneyimli 4 uzmanın, deneyim bazlı test teknikleri konusunda deneyimli 8 uzmanın ve statik test teknikleri konusunda deneyimli 4 uzmanın görüşlerine başvurulmuştur. Aşağıda her bir tekniğin hata bulma oranının ayrıntılı açıklaması ve aynı grup altındaki diğer tekniklerle olan sıralaması yer almaktadır.

3.2.1.Gereksinim Bazlı Test Tekniklerinin Hata Bulma Oranları Açısından Sıralaması

Gereksinimlere dayalı test tekniklerinin hata bulma oranlarını tespit etmek için bu konuda deneyimli 8 uzmana görüşleri sorulduğunda alınan genel cevaplar aşağıdaki gibidir.

3.2.1.1.Denklik Paylarına Ayırma

Aynı tür girdiler ile aynı sonuçların alınabileceği varsayımına dayanan bu test tekniğinde hata bulma oranı oldukça yüksektir. Çünkü bu teknikte geçerli veriler test edilebildiği gibi geçersiz verilerin de geçerliliğinin doğrulanması söz konusu olabilir.

3.2.1.2.Sınır Değer Analizi

Denklik paylarının sınırlarındaki değerleri test ettiğinden ilk tekniğe göre hata bulma oranı daha az olmakla birlikte sınır değerlerde fazla miktarda hata tespit edilebildiği varsayıldığında etkili bir hata bulma oranına sahiptir.

3.2.1.3.Karar Tablosu

Durumların kombinasyonunu ele alan bu teknikte özellikle gereksinimlerdeki eksiklikler ve çelişkiler etkili bir şekilde tespit edilebildiğinden hata bulma oranı oldukça yüksektir.

3.2.1.4.Durum Geçiş Diyagramları

Geçerli ve geçersiz geçişlerin test edildiği bu teknikte diğer tekniklere göre daha az hata tespit edilmektedir.

3.2.1.5.Kullanım Senaryoları

Belirtilen kullanım senaryosu üzerinden test yapılan bu teknikte senaryodaki eksik noktalar rahatlıkla tespit edilebilmektedir.

Teknikleri ikişerli olarak gruplandırıp uzmanlardan tercih yapmaları istendiğinde yaptıkları tercihler aşağıdaki tablodaki gibidir.

Tablo 3.1: Gereksinim Bazlı Test Tasarım Tekniklerinin Hata Bulma Oranları Açısından Tercih Tablosu.

	Denklik Paylarına Ayırma(1)	Sınır Değer Analizi(2)	Karar Tablosu(3)	Durum Geçiş Diyagramı(4)	Kullanım Senaryoları(5)
Denklik Paylarına Ayırma (1)		1	1	1	1
Sınır Değer Analizi (2)	1		3	2	2
Karar Tablosu (3)	1	3		3	3
Durum Geçiş Diyagramı (4)	1	2	3		5
Kullanım Senaryoları (5)	1	2	3	5	

Tablodan da anlaşılabileceği gibi gereksinim bazlı test tekniklerinin hata bulma oranlarına göre çoktan aza doğru sıralaması aşağıdaki gibidir.

1. Denklik Paylarına Ayırma
2. Karar Tablosu
3. Sınır Değer Analizi
4. Kullanım Senaryoları
5. Durum Geçiş Diyagramları

3.2.2.Yapı Bazlı Test Tekniklerinin Hata Bulma Oranları Açısından Sıralaması

Yapı bazlı test tekniklerinin hata bulma oranlarını tespit etmek için bu konuda deneyimli 4 uzmana görüşleri sorulduğunda hepsinden aynı cevap alınmış olup aşağıdaki gibidir.

3.2.2.1.Komut Testi

Bu teknikte komut olan her akış test edilmektedir. Hata bulma oranı karar testinde bulunan hata bulma oranına göre daha azdır. Çünkü komut olmayan akışlar bu testin kapsamında değildir.

3.2.2.2.Karar Testi

Karar testi kod içerisindeki tüm akışları test ettiğinden dolayı hata bulma oranı oldukça fazladır. Ayrıca karar testi komut testini kapsadığından karar testinin hata bulma oranı komut testinin hata bulma oranından oldukça fazladır.

3.2.3.Deneyime Dayalı Test Tekniklerinin Hata Bulma Oranları Açısından Sıralaması

Deneyime dayalı test tekniklerinin hata bulma oranlarını tespit etmek için bu konuda deneyimli 8 uzmana görüşleri sorulduğunda alınan genel cevaplar aşağıdaki gibidir.

3.2.3.1.Keşif Testi

Birçok test tekniğinin tamamlayıcısı olarak görülen bu tekniğin hata bulma oranı oldukça fazladır. Agile modelde en çok kullanılan tekniklerden birisidir.

3.2.3.2.Hata Tahmini

Geçmiş tecrübelerden faydalanılarak hataların çıkabileceği noktaların tahmin edilmesine dayanan bu teknikte hata bulma oranı çok fazla değildir. Çünkü teknik bir dokümana dayandırılmamış ve tespit edilen hatanın tekrarlanabilirlik olasılığı düşüktür.

3.2.3.3.Kontrol Listeleri

Sistemin genel kontrollerini içeren bu teknikte hata bulma oranı keşif testi kadar olmasa da hata tahminleme tekniğinden fazladır.

3.2.3.4.Ataklar

Sistemin güvenliğinin test edildiği bu teknikte hata bulma oranı diğer tekniklere göre daha azdır.

Teknikleri ikişerli olarak gruplandırıp uzmanlardan tercih yapmaları istendiğinde yaptıkları tercihler aşağıdaki tablodaki gibidir.

Tablo 3.2: Deneyime Dayalı Test Tasarım Tekniklerinin Hata Bulma Oranları Açısından Tercih Tablosu.

	Keşif Testi (1)	Hata Tahmini (2)	Kontrol Listeleri (3)	Ataklar (4)
Keşif Testi (1)		1	1	1
Hata Tahmini (2)	1		3	2
Kontrol Listeleri (3)	1	3		3
Ataklar (4)	1	2	3	

Tablodan da anlaşılabileceği gibi deneyime dayalı test tekniklerinin hata bulma oranlarına göre çoktan aza doğru sıralaması aşağıdaki gibidir.

1. Keşif Testi
2. Kontrol Listeleri
3. Hata Tahmini
4. Ataklar

3.2.4.Statik Test Tekniklerinin Hata Bulma Oranları Açısından Sıralaması

Statik test tekniklerinin hata bulma oranlarını tespit etmek için bu konuda deneyimli 4 uzmana görüşleri sorulduğunda alınan genel cevaplar aşağıdaki gibidir.

3.2.4.1.Gayriresmi Gözden Geçirme

Bu teknikte kapsamlı olmayan bir gözden geçirme işlemi yapıldığından resmi tekniklere göre oldukça az hata bulunmaktadır.

3.2.4.2.Üzerinden Geçme

Bu teknikte doküman bir grup tarafından incelendiğinden hata bulma oranı gayriresmi gözden geçirmeye göre daha fazladır.

3.2.4.3.Teknik Gözden Geçirme

Bu teknikte doküman teknik ekip tarafından incelendiğinden dolayı hata bulma oranı diğer iki test tekniğine göre daha fazladır.

3.2.4.4.Teftiş

Gözden geçirme tekniklerinin en resmi olanı kabul edilen teftişte hata bulma oranı oldukça fazladır. Hataların en iyi ve en erken tespit edildiği teknik olarak kabul edilir.

3.2.4.5.Statik Analiz

Kodun çalıştırılmadan gözden geçirilmesi olan statik analizin hata bulma oranı teftiş kadar fazla olmamakla birlikte diğer tekniklerden fazladır.

Teknikleri ikişerli olarak gruplandırıp uzmanlardan tercih yapmaları istendiğinde yaptıkları tercihler aşağıdaki tablodaki gibidir.

Tablo 3.3: Statik Test Tasarım Tekniklerinin Hata Bulma Oranları Açısından Tercih Tablosu.

	Gayri resmi Gözden Geçirme (1)	Üzerinden Geçme (2)	Teknik Gözden Geçirme (3)	Teftiş (4)	Statik Analiz (5)
Gayri resmi Gözden Geçirme (1)		2	3	4	5
Üzerinden Geçme (2)	2		3	4	5
Teknik Gözden Geçirme (3)	3	3		4	5
Teftiş (4)	4	4	4		4
Statik Analiz (5)	5	5	5	4	

Tablodan da anlaşılabileceği gibi statik test tekniklerinin hata bulma oranlarına göre çoktan aza doğru sıralaması aşağıdaki gibidir.

1. Teftiş
2. Statik Analiz
3. Teknik Gözden Geçirme
4. Üzerinden Geçme
5. Gayri Resmi Gözden Geçirme

3.3.TEST TASARIM TEKNİKLERİ UYGULANABİLİRLİK ANALİZİ

Bu bölümde matematiksel modelleme için belirlenen üçüncü parametre olan uygulanabilirlik analizi yer almaktadır. Test tasarım tekniklerinde uzman kişiler için hazırlanan uygulanabilirlik çalışmasının içerik çözümlemesi gerçekleştirilecektir.

Uygulanabilirliği analiz etmek için yine gereksinim bazlı test teknikleri konusunda deneyimli 8 uzmanın, yapı bazlı test teknikleri konusunda deneyimli 4 uzmanın, deneyime dayalı test teknikleri konusunda deneyimli 8 uzmanın ve statik test teknikleri konusunda deneyimli 4 uzmanın görüşlerine başvurulmuştur. Aşağıda her bir tekniğin uygulanabilirlik açısından değerlendirilmesi ve aynı grup altındaki diğer tekniklerle olan sıralaması yer almaktadır. Bu çalışma esnasında ISTQB (International Software Testing Qualifications Board) Advanced Level Test Analyst ve Advanced Level Technical Test Analsyt kaynaklarından yararlanılmıştır.

3.3.1.Gereksinim Bazlı Test Tekniklerinin Uygulanabilirlik Açısından Sıralaması

Gereksinimlere dayalı test tekniklerinin uygulanabilirliğini tespit etmek için bu konuda deneyimli 8 uzmana görüşleri sorulduğunda alınan genel cevaplar ve ISTQB'ye göre tekniklerin uygulanabilirlik açısından değerlendirilmesi aşağıdaki gibidir.

3.3.1.1.Denklik Paylarına Ayırma

Denklik paylarına ayırma tekniği fonksiyonel testler sırasında hemen her geliştirmede uygulanabilir. Çünkü test sırasında her türlü sınıflandırma yapılarak denklik payı oluşturulması mümkün olabilir. Bu teknikte dikkat edilmesi gereken en önemli kriter sınıflandırılan dataların doğru paylara yerleştirilmesidir. Aksi takdirde bu durum yanlış test değerlendirmelerine yol açabilir [30].

3.3.1.2.Sınır Değer Analizi

Sınırlardaki değerlerin test edildiği bu teknik de fonksiyonel testler sırasında hemen her geliştirmede uygulanabilir. Fakat her geliştirmede sınırın test edilmesi gibi bir durum söz konusu olmayabilir. Bu yönüyle uygulanabilirlik açısından denklik paylarına ayırma yöntemine göre kapsamı daha küçüktür. Sınır değer analizindeki en önemli

kriter sınırlardaki değerlerin doğru denklik paylarına ait olduğundan emin olunmasıdır. Aksi takdirde bu durum yanlış test değerlendirmelerine yol açabilir [31].

3.3.1.3.Karar Tablosu

Koşullardan yola çıkılarak oluşturulan bu teknik her fonksiyonel testte uygulanabilir. Çünkü her bir test senaryosu belli koşullara göre oluşturulur. Zaman kısıtı dikkate alındığında bu teknik için indirgenmiş karar tablosu oluşturulurken doğru senaryoların indirgendiğinden emin olunmalıdır [32].

3.3.1.4.Durum Geçiş Diyagramları

Geçiş diyagramları tekniği geçerli ve geçersiz durumların tanımlı olduğu fonksiyonel testlerde uygulanabilir. Kullanım alanı özellikle gömülü sistemlerde fazla olmasına karşın diğer tekniklere oranla daha azdır. Durum geçiş diyagramlarında özellikle dikkat edilmesi gereken kriter geçerli geçişlerin geçerli olduğunun tespit edilmesinin yanında geçersiz geçişlerin de geçersiz olduğundan emin olunmasıdır [33].

3.3.1.5.Kullanım Senaryoları

Kullanım senaryoları her geliştirme için uygulanabilen bir tekniktir. Fakat her durum için kullanım senaryosunun hazırlanması zaman kısıtı açısından mümkün olmayabilir.

Kullanım senaryolarında genellikle ana akışlar ve bu ana akışa alternatif olabilecek akışlar ele alınır [34]. Teknikleri ikişerli olarak gruplandırıp uzmanlardan tercih yapmaları istendiğinde yaptıkları tercihler aşağıdaki tablodaki gibidir.

Tablo 3.4: Gereksinim Bazlı Test Tasarım Tekniklerinin Uygulanabilirlik Açısından Tercih Tablosu.

	Denklik Paylarına Ayırma(1)	Sınır Değer Analizi(2)	Karar Tablosu(3)	Durum Geçiş Diyagramı(4)	Kullanım Senaryoları(5)
Denklik Paylarına Ayırma (1)		1	3	1	5
Sınır Değer Analizi (2)	1		3	2	5
Karar Tablosu (3)	3	3		3	3
Durum Geçiş Diyagramı (4)	1	2	3		5
Kullanım Senaryoları (5)	5	5	3	5	

Tablodan da anlaşılabileceği gibi gereksinim bazlı test tekniklerinin uygulanabilirliklerine göre çoktan aza doğru sıralaması aşağıdaki gibidir.

1.Karar Tablosu

2.Kullanım Senaryoları

3.Denklik Paylarına Ayırma

4.Sınır Değer Analizi

5.Durum Geçiş Diyagramları

3.3.2.Yapı Bazlı Test Tekniklerinin Uygulanabilirlik Açısından Sıralaması

Yapı bazlı test tekniklerinin uygulanabilirliklerini analiz etmek için 4 uzmanın görüşü alındığında ve incelendiğinde iki tekniğin de (komut testi, karar testi) her geliştirme için uygulanabilen teknikler olduğu görülmüştür. Uygulanabilirlik açısından bu iki teknik arasında bir üstünlük durumu olmadığı sonucu çıkarılmıştır.

3.3.3.Deneyime Dayalı Test Tekniklerinin Uygulanabilirlik Açısından Sıralaması

Deneyime dayalı test tekniklerinin uygulanabilirliklerini tespit etmek için bu konuda deneyimli 8 uzmana görüşleri sorulduğunda alınan genel cevaplar aşağıdaki gibidir.

3.3.3.1.Keşif Testi

Keşif testi zaman kısıtı dikkate alındığında uygulaması kritik olabilen bir tekniktir. Diğer tecrübeye dayalı test tekniklerine göre daha uzun süreceğinden uygulanabilirlik açısından daha zayıftır.

3.3.3.2.Hata Tahmini

Hata tahmini uygulaması kolay olan bir tekniktir. Fakat test sırasında oluşturulan senaryoların tekrarlanabilirliği riski açısından kontrol listelerine göre daha az uygulanabilir durumdadır.

3.3.3.3.Kontrol Listeleri

Sistemin genel kontrollerinin yapıldığı kontrol listeleri tekniği her yazılım türünde uygulanabilir. Yazılı kaynağa dayandığı için tekrarlanabilir bir tekniktir. Deneyime dayalı test teknikleri arasında en uygulanabilir teknik olduğu söylenebilir.

3.3.3.4.Ataklar

Sistemin güvenliğinin test edildiği bu tekniğin uygulanması için çeşitli donanımsal altyapı geliştirmeleri gerekebilir. Bu açıdan değerlendirildiğinde deneyime dayalı test teknikleri arasında en az uygulanabilir olan tekniğin bu olduğu söylenebilir.

Teknikleri ikişerli olarak gruplandırıp uzmanlardan tercih yapmaları istendiğinde yaptıkları tercihler aşağıdaki tablodaki gibidir.

Tablo 3.5: Deneyime Dayalı Test Tasarım Tekniklerinin Uygulanabilirlik Açısından Tercih Tablosu.

	Keşif Testi (1)	Hata Tahmini (2)	Kontrol Listeleri (3)	Ataklar (4)
Keşif Testi (1)		2	3	1
Hata Tahmini (2)	2		3	2
Kontrol Listeleri (3)	3	3		3
Ataklar (4)	1	2	3	

Tablodan da anlaşılabileceği gibi deneyime dayalı test tekniklerinin uygulanabilirliklerine göre çoktan aza doğru sıralaması aşağıdaki gibidir.

1.Kontrol Listeleri

2.Hata Tahmini

3.Keşif Testi

4.Ataklar

3.3.4.Statik Test Tekniklerinin Uygulanabilirlik Açısından Sıralaması

Statik test tekniklerinin uygulanabilirliklerini tespit etmek için bu konuda deneyimli 4 uzmana görüşleri sorulduğunda alınan genel cevaplar aşağıdaki gibidir. Statik test teknikleri uygulanabilirlik açısından değerlendirilirken dikkate alınan en önemli kriter zaman olmuştur.

3.3.4.1.Gayriresmi Gözden Geçirme

Statik teknikler içerisinde en uygulanabilir teknik olduğu söylenebilir. Çok fazla efor gerektirmeyen ve hızlıca gerçekleştirilebilen bir sürece sahiptir.

3.3.4.2.Üzerinden Geçme

Genellikle son kullanıcının katılımıyla gerçekleştirilen bu teknik personellerin zaman kısıtı dikkate alındığında gayri resmi gözden geçirme kadar uygulanabilir değildir.

3.3.4.3.Teknik Gözden Geçirme

Teknik bir süreç gerektiren bu teknikte iş ürününün değerlendirilmesi oldukça zaman alan bir süreç olabileceğinden ve bu konuda eğitimli personel azlığından dolayı uygulanabilirlik açısından diğer tekniklere göre daha zayıftır.

3.3.4.4.Teftiş

Resmi süreçlere dayandırılan bu teknikte her bir safha belli bir zamana ihtiyaç duyduğundan zaman kısıtı dikkate alınan projelerde en az uygulanabilen tekniklerden biridir.

3.3.4.5.Statik Analiz

Statik analiz, teknik gözden geçirme gibi teknik bilgi gerektirdiğinden ve bu konuda uzman personel azlığı göz önüne alındığında uygulanabilirlik açısından zayıf olan bir tekniktir.

Teknikleri ikişerli olarak gruplandırıp uzmanlardan tercih yapmaları istendiğinde yaptıkları tercihler aşağıdaki tablodaki gibidir.

Tablo 3.6 : Statik Test Tasarım Tekniklerinin Uygulanabilirlik Açısından Tercih Tablosu.

	Gayriresmi Gözden Geçirme (1)	Üzerinden Geçme (2)	Teknik Gözden Geçirme (3)	Teftiş (4)	Statik Analiz (5)
Gayriresmi Gözden Geçirme (1)		1	1	1	1
Üzerinden Geçme (2)	1		2	2	2
Teknik Gözden Geçirme (3)	1	2		4	5
Teftiş (4)	1	2	4		5
Statik Analiz (5)	1	2	5	5	

Tablodan da anlaşılabileceği gibi statik test tekniklerinin hata bulma oranlarına göre çoktan aza doğru sıralaması aşağıdaki gibidir.

1.Gayriresmi Gözden Geçirme

2.Üzerinden Geçme

3.Statik Analiz

3.Teftiş

4.Teknik Gözden Geçirme

3.4.TEST TASARIM TEKNİKLERİNİN MATEMATİKSEL MODELLEMESİ

Önceki bölümlerde yazılım testi tasarım teknikleri aşağıdaki konular açısından analiz edilmişti:

- Kullanım sıklığı
- Hata bulma oranı
- Uygulanabilirlik

Bu bölümde ise analizi yapılan bu konularda elde edilen verilerden ve Bayes teoreminden yararlanılarak karar verme fonksiyonları oluşturulacaktır.

Önceki bölümlerde test tasarım tekniklerinin analiz edildiği kullanım sıklığı, hata bulma oranı ve uygulanabilirlik alt kümelerinin bulunduğu büyük kümeye A diyelim. ‘X’ rastgele değişkeninin veya kümesinin A kümesinin özelliklerini taşıdığını varsayalım. O halde P(X) yani X rastgele değişkeninin A kümesinin elemanı olma olasılığı hesaplanabilir.

Kullanım sıklığı, hata bulma oranı ve uygulanabilirlik gibi analiz konularının her birinin alt küme olduğunu varsayarsak bu alt kümeler (A_1, A_2,...,A_n) olarak gösterilebilir. ‘n’; test tasarım tekniklerinin incelendiği analiz başlık sayısını temsil etmektedir. (X_1, X_2,...,X_n) rastgele değişkenleri sırasıyla (A_1, A_2,...,A_n) kümelerinin elemanları olduğunu ve bu değişkenlerin yalnızca bu kümelere ait (birbirini dışlayan) olduğunu varsayarsak X kümesi;

$$X = (X \cap X_1) \cup (X \cap X_2) \cup .. \cup (X \cap X_N) \quad (1)$$

olarak ifade edilebilir [35].O halde X olaylarının olasılığı;

$$P(X) = P(X \cap X_1) + P(X \cap X_2) + \cdots + P(X \cap X_k) = \sum_{k=1}^{N} P(X \cap X_k) \quad (2)$$

olarak ifade edilir [36].

Bayes teoremine dayanarak X_k olayı oluşurken X olayı olabilme koşulu;

$$P(X|X_k) = \frac{P(X \cap X_k)}{P(X_k)} \quad (3)$$

olarak ifade edilir [37].

Buradan ;

$$P(X \cap X_k) = P(X|X_k)P(X_k) \quad (4)$$

ifadesi elde edilebilir.

Bu durumda X olayının olasılığı için;

$$P(X) = P(X|X_1)P(X_1) + P(X|X_2)P(X_2) + \cdots + P(X|X_n)P(X_n) = \sum_{k=1}^{N} P(X|X_k)P(X_k) \quad (5)$$

ifadesi kullanılabilir.

$P(X|X_k)$koşullu olasılığını W_k olarak tanımlayalım. Bu durumda olasılık fonksiyonu aşağıdaki gibi olur.

$$P(X) = \sum_{k=1}^{N} W_k P(X_k) \quad (6)$$

Yazılım testi tasarım tekniklerinin matematiksel modellemesinde yukarıdaki fonksiyon bir karar verme fonksiyonu gibi kullanılarak her bir tasarım alanı grubu için en uygun olan teknik belirlenebilir.

$$F(X) = \sum_{k=1}^{N} W_k X_k \quad (7)$$

Bu fonksiyonda F(X) (Bayes Teoreminde P(X) olarak ele alınır) ; her bir tasarım tekniği için karar fonksiyon değerini ifade etmektedir. W_k değerleri her bir tasarım tekniği için ele alınan analiz grubunun ağırlık değerini temsil etmektedir. Örneğin; W_1 değeri bu çalışma için kullanım sıklığı ağırlık değeri olarak hesaplanmıştır. X_k değerleri de her bir

test tasarım tekniği için ilgili analiz sonucunda elde edilen rasyonel değeri temsil etmektedir.

Karar verme fonksiyonu uygulanırken X olaylar kümesinin alt kümeleri analiz konuları olarak ele alınmıştır.

Kullanım Sıklığı: Yazılım testi konusunda uzman 65 kişiye anket yapılarak profil bilgilerinin, görev aldıkları firmaların özelliklerinin analizinin yanı sıra test tasarım tekniklerinin kullanım sıklığı ve uzmanlar için öneminin tespit edilmesi hedeflenmiştir. Bu çalışmada uzmanlara danışılarak bu özelliğin ağırlığı %25 olarak belirlenmiştir.

Hata Bulma Oranı: Her tasarım tekniği tipinden belli sayıda uzmanın görüşlerine dayanarak hazırlanan çalışmada test tasarım tekniklerinin ürün üzerindeki hata bulma etkisinin tespit edilmesi hedeflenmiştir. Bu çalışmada uzmanlara danışılarak bu özelliğin ağırlığı % 40 olarak belirlenmiştir.

Uygulanabilirlik: Her bir test tasarım tekniğinin uygulanabilirliğini tespit etmek için ISTQB kaynaklarından yararlanılmış ve uzmanların görüşlerine başvurulmuştur. Bu çalışmada uzmanlara danışılarak bu özelliğin ağırlığı %35 olarakbelirlenmiştir.

4. BULGULAR

Daha önceki bölümlerde Bayes teoreminden faydalanılarak karar verme fonksiyonunun oluşturulmasından bahsedilmişti. Bu bölümde her bir test tasarım tekniği kümesinde karar fonksiyonu ve vaka analizi uygulanarak optimum fayda sağlayan test tasarım tekniği belirlenecektir.

4.1.GEREKSİNİM BAZLI TEST TEKNİKLERİ İÇİN KARAR VERME VE VAKA ANALİZİ

Önceki bölümlerde gereksinim bazlı test teknikleri kullanım sıklığı, hata bulma oranı ve uygulanabilirlik açısından analiz edilmiş ve aşağıdaki sonuçlar alınmıştır.

Tablo 4.1: Gereksinim Bazlı Test Tasarım Teknikleri Parametre Değer Tablosu.

	Kullanım Sıklığı	Hata Bulma Oranı	Uygulanabilirlik
Denklik Paylarına Ayırma	7,03	5	3
Sınır Değer Analizi	8,25	3	2
Karar Tablosu	7,98	4	5
Geçiş Diyagramları	6,62	1	1
Kullanım Senaryosu	7,8	2	4
Ağırlıklar	25	40	35

Tablodaki değerleri normalize etmek için kullanım sıklığı kolonundaki bütün değerler en büyük kullanım sıklığı oranına sahip olan 8,25 ‘e; hata bulma oranı kolonundaki bütün değerler en fazla hata bulma oranına sahip olan 5’e ve uygulanabilirlik kolonundaki bütün değerler de en fazla uygulanabilirlik oranına sahip olan 5’ e bölünmüştür. Ağırlıkların da toplam ağırlığa bölünmesiyle aşağıdaki tablo elde edilmiştir.

Tablo 4.2: Gereksinim Bazlı Test Tasarım Teknikleri Parametre Normalize Değer Tablosu.

	Kullanım Sıklığı	Hata Bulma Oranı	Uygulanabilirlik
Denklik Paylarına Ayırma (1)	0,8521	1	0,6
Sınır Değer Analizi (2)	1	0,6	0,4
Karar Tablosu (3)	0,9672	0,8	1
Geçiş Diyagramları (4)	0,8024	0,2	0,2
Kullanım Senaryosu (5)	0,9454	0,4	0,8
Ağırlıklar	0,25	0,4	0,35

Gereksinim bazlı her bir test tekniği için karar fonksiyon değerleri aşağıdaki gibidir.

F(1) = 0,25 x **0,8521** + 0,4 x **1** + 0,35 x **0,6** = 0,2130 + 0,4 + 0,21 = **0,823**

F(2) = 0,25 x **1** + 0,4 x **0,6** + 0,35 x **0,4** = 0,25 + 0,24 + 0,14 = **0,63**

F(3) = 0,25 x **0,9672** + 0,4 x **0,8** + 0,35 x **1** = 0,2418 + 0,32 + 0,35 = **0,9118**

F(4) = 0,25 x **0,8024** + 0,4 x **0,2** + 0,35 x **0,2** = 0,2006 + 0,08 + 0,07 = **0,3506**

F(5) = 0,25 x **0,9454** + 0,4 x **0,4** + 0,35 x **0,8** = 0,2363 + 0,16 + 0,28 = **0,6763**

Bu sonuçlara bakılarak gereksinim bazlı test tasarım teknikleri arasında karar tablosu tekniğinin en uygun teknik olduğu görülmüştür.

Tablo 4.3: Gereksinim Bazlı Test Tasarım Teknikleri Karar Değer Tablosu.

Test Tasarım Tekniği	**Karar Fonksiyon Değeri**
Denklik Paylarına Ayırma	0,823
Sınır Değer Analizi	0,63
Karar Tablosu	**0,9118**
Geçiş Diyagramları	0,3506
Kullanım Senaryosu	0,6763

4.2.YAPI BAZLI TEST TEKNİKLERİ İÇİN KARAR VERME VE VAKA ANALİZİ

Önceki bölümlerde yapı bazlı test teknikleri kullanım sıklığı, hata bulma oranı ve uygulanabilirlik açısından analiz edilmiş ve aşağıdaki sonuçlar alınmıştır.

Tablo 4.4: Yapı Bazlı Test Tasarım Teknikleri Parametre Değer Tablosu.

	Kullanım Sıklığı	Hata Bulma Etkisi	Uygulanabilirlik
Komut Testi	6,61	1	1
Karar Testi	6,78	2	1
Ağırlıklar	0,25	0,4	0,35

Tablodaki değerleri normalize etmek için kullanım sıklığı kolonundaki bütün değerler en büyük kullanım sıklığı oranına sahip olan 6,78 ‘e; hata bulma oranı kolonundaki bütün değerler en fazla hata bulma oranına sahip olan 2’ye bölünmüştür. Uygulanabilirlik kolonunda değerler aynı olduğundan normalizasyona gerek durulmamıştır. Normalize edilmiş olan veriler aşağıdaki gibidir.

Tablo 4.5: Yapı Bazlı Test Tasarım Teknikleri Parametre Normalize Değer Tablosu.

	Kullanım Sıklığı	Hata Bulma Etkisi	Uygulanabilirlik
Komut Testi (1)	0,9749	0,5	1
Karar Testi (2)	1	1	1
Ağırlıklar	0,25	0,4	0,35

Yapı bazlı her bir test tekniği için karar fonksiyon değerleri aşağıdaki gibidir.

F(1) = 0,25 x **0,9749** + 0,4 x **0,5** + 0,35 x **1** = 0,2437 + 0,02 + 0,35 = **0,6137**

F(2) = 0,25 x **1** + 0,4 x **1** + 0,35 x **1**= 0,25 + 0,4 + 0, 35 = **1**

Bu sonuçlara bakılarak yapı bazlı test tasarım teknikleri arasında karar testi tekniğinin en uygun teknik olduğu görülmüştür.

Tablo 4.6: Yapı Bazlı Test Tasarım Teknikleri Karar Değer Tablosu.

Test Tasarım Tekniği	**Karar Fonksiyon Değeri**
Komut Testi	0,6137
Karar Testi	**1**

4.3.DENEYİME DAYALI TEST TEKNİKLERİ İÇİN KARAR VERME VE VAKA ANALİZİ

Önceki bölümlerde deneyime dayalı test teknikleri kullanım sıklığı, hata bulma oranı ve uygulanabilirlik açısından analiz edilmiş ve aşağıdaki sonuçlar alınmıştır.

Tablo 4.7: Deneyime Dayalı Test Tasarım Teknikleri Parametre Değer Tablosu.

	Kullanım Sıklığı	Hata Bulma Etkisi	Uygulanabilirlik
Keşif Testi	7,3	4	2
Hata Tahmini	7,7	2	3
Kontrol Listesi	8,02	3	4
Ataklar	5,33	1	1
Ağırlıklar	0,25	0,4	0,35

Tablodaki değerleri normalize etmek için kullanım sıklığı kolonundaki bütün değerler en büyük kullanım sıklığı oranına sahip olan 8,02 ‘ye; hata bulma oranı kolonundaki bütün değerler en fazla hata bulma oranına sahip olan 4’e ve uygulanabilirlik kolonundaki bütün değerler de en fazla uygulanabilirlik oranına sahip olan 4’ e bölünmüştür. Ağırlıkların da toplam ağırlığa bölünmesiyle aşağıdaki tablo elde edilmiştir.

Tablo 4.8: Deneyime Dayalı Test Tasarım Teknikleri Parametre Normalize Değer Tablosu.

	Kullanım Sıklığı	Hata Bulma Etkisi	Uygulanabilirlik
Keşif Testi (1)	0,9102	1	0,5
Hata Tahminleme (2)	0,96	0,5	0,75
Kontrol Listesi (3)	1	0,75	1
Ataklar (4)	0,6645	0,25	0,25
Ağırlıklar	0,25	0,4	0,35

Deneyime dayalı her bir test tekniği için karar fonksiyon değerleri aşağıdaki gibidir.

F(1) = 0,25 x **0,9102** + 0,4 x **1** + 0,35 x **0,5** = 0,2275 + 0,4 + 0,175 = **0,8025**

F(2) = 0,25 x **0,96** + 0,4 x **0,5** + 0,35 x **0,75** = 0,24 + 0,02 + 0,2625 = **0,5225**

F(3) = 0,25 x **1** + 0,4 x **0,75** + 0,35 x **1** = 0,25 + 0,3 + 0,35 = **0,9**

F(4) = 0,25 x **0,6645** + 0,4 x **0,25** + 0,35 x **0,25** = 0,1661 + 0,1 + 0,0875 = **0,3536**

Bu sonuçlara bakılarak deneyime dayalı test tasarım teknikleri arasında kontrol listesi tekniğinin en uygun teknik olduğu görülmüştür.

Tablo 4.9: Deneyime Dayalı Test Tasarım Teknikleri Karar Değer Tablosu.

Test Tasarım Tekniği	**Karar Fonksiyon Değeri**
Keşif Testi	0,8025
Hata Tahminleme	0,5225
Kontrol Listesi	**0,9**
Ataklar	0,3536

4.4.STATİK TEST TEKNİKLERİ İÇİN KARAR VERME VE VAKA ANALİZİ

Önceki bölümlerde statik test teknikleri kullanım sıklığı, hata bulma oranı ve uygulanabilirlik açısından analiz edilmiş ve aşağıdaki sonuçlar alınmıştır.

Tablo 4.10: Statik Test Tasarım Teknikleri Parametre Değer Tablosu.

	Kullanım Sıklığı	Hata Bulma Oranı	Uygulanabilirlik
Gayriresmi Gözden Geçirme	5,93	1	5
Üzerinden Geçme	6,27	2	4
Teknik Gözden Geçirme	6,03	3	1
Teftiş	5,56	5	2
Statik Analiz	5,79	4	3
Ağırlıklar	0,25	0,4	0,35

Tablodaki değerleri normalize etmek için kullanım sıklığı kolonundaki bütün değerler en büyük kullanım sıklığı oranına sahip olan 6,27 ‘ye; hata bulma oranı kolonundaki

bütün değerler en fazla hata bulma oranına sahip olan 5'e ve uygulanabilirlik kolonundaki bütün değerler de en fazla uygulanabilirlik oranına sahip olan 5' e bölünmüştür. Ağırlıkların da toplam ağırlığa bölünmesiyle aşağıdaki tablo elde edilmiştir.

Tablo 4.11: Statik Test Tasarım Teknikleri Parametre Normalize Değer Tablosu.

	Kullanım Sıklığı	Hata Bulma Oranı	Uygulanabilirlik
Gayriresmi Gözden Geçirme (1)	0,9457	0,2	1
Üzerinden Geçme (2)	1	0,4	0,8
Teknik Gözden Geçirme (3)	0,9617	0,6	0,2
Teftiş (4)	0,8867	1	0,4
Statik Analiz (5)	0,9234	0,8	0,6
Ağırlıklar	0,25	0,4	0,35

Statik test tekniklerinin herbiri için karar fonksiyon değerleri aşağıdaki gibidir.

F(1) = 0,25 x **0,9457** + 0,4 x **0,2** + 0,35 x **1** = 0,2364 + 0,08 + 0,35 = **0,6664**

F(2) = 0,25 x **1** + 0,4 x **0,4** + 0,35 x **0,8** = 0,25 + 0,16 + 0,28 = **0,69**

F(3) = 0,25 x **0,9617** + 0,4 x **0,6** + 0,35 x **0,2** = 0,2404 + 0,24 + 0,07 = **0,5504**

F(4) = 0,25 x **0,8867** + 0,4 x **1** + 0,35 x **0,4** = 0,2216 + 0,4 + 0,14 = **0,7616**

F(5) = 0,25 x **0,9234** + 0,4 x **0,8** + 0,35 x **0,6** = 0,2308 + 0,32 + 0,21 = **0,7608**

Bu sonuçlara bakılarak statik test teknikleri arasında teftiş tekniğinin en uygun teknik olduğu görülmüştür.

Tablo 4.12: Statik Test Tasarım Teknikleri Karar Değer Tablosu.

Test Tasarım Tekniği	**Karar Fonksiyon Değeri**
Gayriresmi Gözden Geçirme	0,6664
Üzerinden Geçme	0,69
Teknik Gözden Geçirme	0,5504
Teftiş	**0,7616**
Statik Analiz	0,7608

5. TARTIŞMA VE SONUÇ

Test tasarım teknikleri bu çalışma ile analiz edilmiş ve dört grup altında sınıflandırılarak her bir gruptaki en uygun teknik matematiksel modelleme ile tespit edilmiştir.

Gereksinim bazlı test tekniklerinde en uygun teknik karar tablosu tekniği olarak tespit edilmiştir. Bu tespit öngörülebilir bir sonuçtur. Çünkü karar tablosu kapsamı ve işlevselliği ile diğer tasarım tekniklerinden daha üstündür.

Yapı bazlı test tekniklerinde en uygun teknik karar testi tekniği olarak tespit edilmiştir. Bu tespit öngörülebilir bir sonuçtur. Çünkü yapı bazlı test teknikleri başlığı altında bulunan diğer teknik olan komut testi tekniği; karar testinin her bakımdan bir alt kümesidir. Karar testi komut testini kapsar.

Deneyime dayalı test tekniklerinde en uygun teknik kontrol listesi tekniği olarak tespit edilmiştir. Bu tespit yapılan çalışmalar öncesinde tespit edilememiştir. En uygun tekniğin keşif testi olduğu düşünülmüştür. Fakat keşif testinin kullanım sıklığı ve uygulanabilirlik açısından kontrol listesi tekniğinden daha zayıf olduğu görülmüştür.

Statik test tekniklerinde en uygun teknik teftiş tekniği olarak tespit edilmiştir. Bu tespit öngörülebilir bir sonuçtur. Çünkü teftiş birçok yönden diğer test tekniklerine göre daha çok tercih edilen ve statik test teknikleri arasında en resmi sürece sahip olan tekniktir.

KAYNAKLAR

[1]. TDK, 2016, *Yazılım*, https://tr.wikipedia.org/wiki/Yaz%C4%B1l%C4%B1m, [Ziyaret Tarihi: 5 Şubat 2016]

[2]. Kızmaz, V., 2012, *Yazılım Yaşam Döngüsü Nedir*, http://www.ugurkizmaz.com/YazilimMakale-1825-Yazilim-Yasam-Dongusu-Nedir.aspx, [Ziyaret Tarihi: 11 Mart 2016]

[3]. Han, A., 2014, *Yazılım Geliştirme Yaşam Döngüsü*, http://www.ele ktrikport.com/teknik-kutuphane/yazilim-gelistirme-yasam-dongusu/11471#ad-image-7, [Ziyaret Tarihi: 18 Mart 2016]

[4]. İTÜ BİDB, 2013, *Yazılım Testi ve Test Süreçleri*, http://bidb.itu.edu.tr/seyirdefteri/blog/2013/09/08/yaz%C4%B1l%C4%B1m-testi-ve-test-s%C3%Bcre%C3%A7leri, [Ziyaret Tarihi: 7 Nisan 2016]

[5]. Vikipedi, 2016, Yazılım Testi,https://tr.wikipedia.org/wiki/Yaz%C4%B1l%C4%B1m _test_etme , [Ziyaret Tarihi: 7 Nisan 2016]

[6]. TTB, 2014, *Sertifikalı Test Uzmanı Temel Seviye Ders Programı,* Yazılım Test ve Kalite Derneği, Bölüm 1, Turkey, 15.

[7]. Tassey, G., 2002, *The Economic Impacts of Inadequate Infrastructurefor Software Testing*, Planning Report 02(3), National Institute of Standards and Technology (NIST)

[8]. TTB, 2014, *Sertifikalı Test Uzmanı Temel Seviye Ders Programı,* Yazılım Test ve Kalite Derneği, Bölüm 1, Turkey, 16.

[9]. Black, R., *ISTQB Advanced Level Syllabus Test Manager*, ISTQB, Chapter 1, 9.

[10]. Black, R., *ISTQB Advanced Level Syllabus Test Manager*, ISTQB, Chapter 1, 11.

[11]. TTB, 2015, *Sertifikalı Test Uzmanı İleri Seviye Ders Programı Test Analisti,* Yazılım Test ve Kalite Derneği, Bölüm 1, Turkey, 17.

[12]. TTB, 2016, *ISTQB Yazılım Testi Terimler Sözlüğü,* Yazılım Test ve Kalite Derneği, Turkey

[13]. TTB, 2015, *Sertifikalı Test Uzmanı İleri Seviye Ders Programı Test Analisti,* Yazılım Test ve Kalite Derneği, Bölüm 1, Turkey, 21.

[14]. TTB, 2014, *Sertifikalı Test Uzmanı Temel Seviye Ders Programı,* Yazılım Test ve Kalite Derneği, Bölüm 1, Turkey, 25.

[15]. TTB, 2014, *Sertifikalı Test Uzmanı Temel Seviye Ders Programı,* Yazılım Test ve Kalite Derneği, Bölüm 1, Turkey, 26.

[16]. TTB, 2014, *Sertifikalı Test Uzmanı Temel Seviye Ders Programı,* Yazılım Test ve Kalite Derneği, Bölüm 1, Turkey, 27.

[17]. Vikipedi, 2016, *ISO/IEC 9126,* https://en.wikipedia.org/wiki/ISO/IEC_9126 ,[Ziyaret Tarihi: 14 Haziran 2016]

[18]. TTB, 2014, *Sertifikalı Test Uzmanı Temel Seviye Ders Programı,* Yazılım Test ve Kalite Derneği, Bölüm 5, Turkey, 54.

[19]. Black, R., *ISTQB Advanced Level Syllabus Test Manager,* ISTQB, Chapter 2, 25.

[20]. IEEE Computer Society, 2009, *IEEE Standart Classification for Software Anomalies (Standart 1044TM),* IEEE, Chapter 3, Turkey, 54.

[21]. TTB, 2016, *ISTQB* Yazılım Testi Terimler Sözlüğü, Yazılım Test ve Kalite Derneği, Turkey

[22]. TTB, 2014, *Sertifikalı Test Uzmanı Temel Seviye Ders Programı,* Yazılım Test ve Kalite Derneği, Bölüm 5, Turkey, 51.

[23]. TTB, 2014, *Sertifikalı Test Uzmanı Temel Seviye Ders Programı,* Yazılım Test ve Kalite Derneği, Bölüm 3, Turkey, 33.

[24]. TTB, 2014, *Sertifikalı Test Uzmanı Temel Seviye Ders Programı,* Yazılım Test ve Kalite Derneği, Bölüm 3, Turkey, 35.

[25]. Kalay, A., 2009, *Statik Kod Analizinde İncelenen Konular*, Statik Kod Analizinin Yazılım Geliştirme Sürecindeki Yeri ve Yazılım Kalitesine Etkisi, 8-10 Ekim 2009 Yıldız Teknik Üniversitesi Oditoryumu, İstanbul, 212.

[26]. Swansea University, 2012, *Equivalence Class Partitioning*, http://www.cs.swan.ac.uk/~csmarkus/CS339/presentations/20061201_Davies_Equivalence_Class_Testing.pdf , [Ziyaret Tarihi: 18 Mart 2016]

[27]. TTB, 2015, *Sertifikalı Test Uzmanı İleri Seviye Ders Programı Test Analisti,* Yazılım Test ve Kalite Derneği, Bölüm 3, Turkey, 32.

[28]. Arnicae, V., 2009, *Complexity of Boundary Value Testing Methods*, Complexity of Equivalence Class and Boundary Value Testing Methods, 751, (80-101).

[29]. TTB, 2015, *Sertifikalı Test Uzmanı İleri Seviye Ders Programı Test Analisti,* Yazılım Test ve Kalite Derneği, Bölüm 3, Turkey, 35.

[30]. TTB, 2015, *Sertifikalı Test Uzmanı İleri Seviye Ders Programı Test Analisti,* Yazılım Test ve Kalite Derneği, Bölüm 3, Turkey, 32.

[31]. TTB, 2015, *Sertifikalı Test Uzmanı İleri Seviye Ders Programı Test Analisti,* Yazılım Test ve Kalite Derneği, Bölüm 3, Turkey, 32.

[32]. TTB, 2015, *Sertifikalı Test Uzmanı İleri Seviye Ders Programı Test Analisti,* Yazılım Test ve Kalite Derneği, Bölüm 3, Turkey, 33.

[33]. TTB, 2015, *Sertifikalı Test Uzmanı İleri Seviye Ders Programı Test Analisti,* Yazılım Test ve Kalite Derneği, Bölüm 3, Turkey, 35.

[34]. TTB, 2015, *Sertifikalı Test Uzmanı İleri Seviye Ders Programı Test Analisti,* Yazılım Test ve Kalite Derneği, Bölüm 3, Turkey, 37.

[35]. Ross, S.M., 2012, *Elements of Probability*, *Probability and Statistics for Engineers and Scientists,* Chapter 3, Academic Press, 58

[36]. Ross, S.M., 2012, *Elements of Probability*, *Probability and Statistics for Engineers and Scientists,* Chapter 3, Academic Press, 59

[37]. Ross, S.M., 2012, *Elements of Probability*, *Probability and Statistics for Engineers and Scientists,* Chapter 3, Academic Press, 72

ÖZGEÇMİŞ

Kişisel Bilgiler	
Adı Soyadı	Asım Kerem Hancı
Doğum Yeri	Bakırköy/İstanbul
Doğum Tarihi	28.07.1988
Uyruğu	T.C.
Telefon	0545 917 52 04
E-Posta Adresi	kerem_hanci@hotmail.com
Web Adresi	-



Eğitim Bilgileri	
Lisans	
Üniversite	Yıldız Teknik Üniversitesi
Fakülte	Kimya Metalurji Fakültesi
Bölümü	Matematik Mühendisliği
Mezuniyet Yılı	2012

Yüksek Lisans	
Üniversite	İstanbul Üniversitesi
Enstitü Adı	Fen Bilimleri Enstitüsü
Anabilim Dalı	Enformatik Anabilim Dalı
Programı	Enformatik
Mezuniyet Tarihi	01.12.2017