

KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

KULLANICI ODAKLI DAVRANIŞ GÜDÜMLÜ YAZILIM
GELİŞTİRME MODELİ

NAİLE DAMLA GÜNCAN

KOCAELİ 2016

**KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

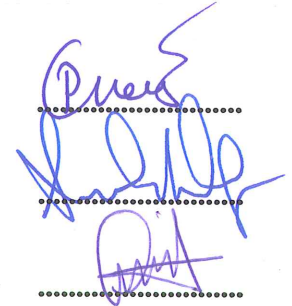
**BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI**

YÜKSEK LİSANS TEZİ

**KULLANICI ODAKLI DAVRANIŞ GÜDÜMLÜ YAZILIM
GELİŞTİRME MODELİ**

NAİLE DAMLA GÜNCAN

Yrd.Doç.Dr. Pınar ONAY DURDU
Danışman, Kocaeli Üniversitesi
Prof.Dr. Semih BİLGİN
Jüri Üyesi, Yeditepe Üniversitesi
Yrd.Doç.Dr. Davut İNCEBACAK
Jüri Üyesi, Kocaeli Üniversitesi



Tezin Savunulduğu Tarih: 16.06.2016

ÖNSÖZ VE TEŞEKKÜR

Bu çalışmada Yazılım Mühendisliği alanından çevik metodolojilerin prensipleri, İnsan Bilgisayar Etkileşimi alanından ise kullanıcı merkezli tasarımının iyi pratikleri entegre edilerek yazılım yaşam döngüsü modeli önerilmektedir. Süreç geliştirilirken anket ve görüşmeler yapılarak, bu konuda tecrübesi olan kişilerden bilgiler alınmış ve döngünün bir bölümünün değerlendirilmesi yapılmıştır.

Yüksek lisans eğitimim boyunca bana her türlü desteği gösteren, sorunlarımı dinleyip çözüm yolları arayan, tezimin her aşamasında yardımını esirgemeyen sevgili tez danışmanım Yrd Doç Dr Pınar ONAY DURDU ve tez savunmamda yer alan jüri üyelerine teşekkürlerimi bir borç bilirim. Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümüne teşekkürlerimi sunarım.

Sonsuz desteği, sabrı ve yardımlarından dolayı sevgili eşim Muhammet Alican Güncan'a ve desteklerini esirgemeyen değerli ailesine, tüm hayatım boyunca her zaman destekçim olup beni sabırla dinleyen ve beni her daim yüreklendiren canım annem Nagihan Coşkan, canım babam Necdet Coşkan ve değerli kardeşim Bedirhan Coşkan'a sonsuz teşekkürlerimi iletmek isterim.

Temmuz – 2016

Naile Damla GÜNCAN

İÇİNDEKİLER

ÖNSÖZ VE TEŞEKKÜR.....	i
İÇİNDEKİLER	ii
ŞEKİLLER DİZİNİ.....	v
TABLOLAR DİZİNİ	vi
SİMGELER VE KISALTMALAR DİZİNİ	vii
ÖZET.....	viii
ABSTRACT.....	ix
GİRİŞ	1
1. LİTERATÜR TARAMASI.....	4
1.1. Yazılım Mühendisliği ve Yazılım Geliştirme Modelleri	4
1.1.1. Uyarlanabilir yazılım geliştirme (ASD)	6
1.1.2. Davranış odaklı geliştirme (BDD)	6
1.2. İnsan Bilgisayar Etkileşimi ve Kullanılabilirlik	8
1.3. Kullanılabilirliğin Yazılım Geliştirme Organizasyonlarında Uygulanması.....	8
1.4. Kullanılabilirliğin Yazılım Yaşam Döngüsüne Entegrasyonu ile İlişkili Çalışmalar	13
2. ARAŞTIRMA YÖNTEMİ.....	18
2.1. Araştırma Stratejisi.....	18
2.2. Araştırma Tasarımı.....	19
2.3. Katılımcılar.....	21
2.4. Değerlendirme Uygulaması.....	23
2.5. Veri Toplama Araçları.....	24
2.6. Veri Analizi	26
2.7. Çalışmanın Güvenilirliği	28
3. ANALİZ VE DEĞERLENDİRME BULGULARI	30
3.1. Birinci Faz: Anket Çalışmasının Bulguları	30
3.1.1. Katılımcı ve organizasyon bilgileri	30
3.1.2. Kullanılabilirlik kavramının anlaşılması ve uygulanması.....	33
3.1.3. Kullanılabilirlik değerlendirmelerinin gerçekleştirilmesi	35
3.1.4. Yazılım geliştirme profesyonellerinin kullanılabilirlik algıları.....	36
3.1.5. Anket Değerlendirmesi	37
3.2. Birinci Faz: Görüşme Bulguları	38
3.2.1. Kullanılabilirlik kavramının anlaşılması ve uygulanması.....	38
3.2.1.1. İBE ya da kullanılabilirlik konularıyla ilgili personel.....	38
3.2.1.2. Kullanılabilirliğin sağlanmasına verilen önem	39
3.2.1.3. Organizasyonda uygulanan yazılım yaşam döngüsü	40
3.2.1.4. Kullanılabilirliğin yaşam döngüsüne entegrasyonu	40
3.2.1.5. Uygulanan modelleme yaklaşımları.....	41
3.2.2. Kullanılabilirlik değerlendirmelerinin gerçekleştirilmesi	41
3.2.2.1. Kullanılabilirlik gereksinimlerinin toplanması	41

3.2.2.2. Kullanılabilirlik yöntemlerini uygulama	42
3.2.2.3. Kullanılabilirlik değerlendirme sonuçları	43
3.2.3. Yazılım geliştirme profesyonellerinin kullanılabilirlik algıları.....	43
3.2.3.1. Kullanılabilirliğin avantajları ve dezavantajları.....	43
3.2.3.2. Yazılım döngüsüne kullanılabilirlik entegrasyonu	44
3.2.3.3. Kullanılabilirlik entegrasyonunda aktiviteler.....	44
3.2.4. Görüşmelerin Değerlendirilmesi	45
3.3. Modelin İlk Sürümü	46
3.3.1. Analiz ve Tasarım	47
3.3.1.1. Ön analiz aşaması (pre-analysis)	48
3.3.1.2. Prototiplemeye dayalı erken test	48
3.3.2. Gerçekleştirme	49
3.3.3. Değerlendirme	50
3.3.4. Dağıtım.....	50
3.3.5. Yinelemeler ve UYG.....	50
3.4. İhtiyaç Modelinin Değerlendirilmesi ve Bulguları	51
3.4.1. Kullanıcı deneyimi konusunda farkındalık	51
3.4.2. Çevik yöntemler konusunda tecrübeler	51
3.4.2.1. Kullanılan çevik yöntemler	51
3.4.2.2. Çevik yaklaşımlardan TOG/DOG (TDD/BDD) hakkında tecrübe	52
3.4.3. KOD analiz ve tasarım safhası	52
3.4.3.1. Ders projesinde gerçekleştirilen süreç ve takım çalışması.....	52
3.4.3.2. DOG (BDD)'un sürece katkısı.....	53
3.4.4. DOG (BDD) ve kullanıcı odaklı tasarım.....	53
3.4.5. Görüşmelerin değerlendirilmesi	54
3.5. İyileştirmeler	54
4. KOD YAZILIM GELİŞTİRME MODELİ'NİN SON SÜRÜMÜ	56
4.1. KOD Yazılım Geliştirme ve KOD Senaryosu	57
4.2. KOD Yazılım Geliştirme ve Fazlar.....	60
4.2.1. Yineleme 0	61
4.2.1.1. Yineleme 0 fazının girdi, çıktı ve amaçları	62
4.2.2. Analiz ve Tasarım	63
4.2.2.1. Ön analiz aşaması (Pre-analiz).....	63
4.2.2.2. Prototiplemeye dayalı erken test	64
4.2.2.3. Analiz ve tasarım fazının girdi, çıktı ve amaçları	65
4.2.3. Gerçekleştirme	65
4.2.3.1. Gerçekleştirme fazının girdi, çıktı ve amaçları	66
4.2.4. Değerlendirme	67
4.2.4.1. Değerlendirme fazının girdi, çıktı ve amaçları	69
4.2.5. Dağıtım (Deployment)	70
4.2.5.1. Dağıtım fazının girdi, çıktı ve amaçları	71
4.3. KOD Yazılım Geliştirme ve Yinelemeleri	71
4.4. KOD Yazılım Geliştirme ve Roller ve Sorumluluklar	72
4.4.1. Müşteriler ve kullanıcılar	72
4.4.2. Analist ekibi	73
4.4.3. Test ekibi	74
4.4.4. UX uzmanı	75

4.4.5. Yazılımcılar	76
5. SONUÇLAR	77
5.1. Yapılan Çalışmanın Özeti	77
5.2. KOD Yazılım Geliştirme Modeli Özeti	78
5.3. KOD Yazılım Geliştirmenin Katkıları	79
5.4. Çalışmanın Kısıtları.....	80
5.5. Gelecek Çalışmalar.....	80
KAYNAKLAR	82
EKLER.....	87
KİŞİSEL YAYIN VE ESERLER	113
ÖZGEÇMİŞ	114



ŞEKİLLER DİZİNİ

Şekil 1.1. DOG Yaklaşımı	8
Şekil 2.1. Çeşitleme Tasarımı	19
Şekil 2.2. Araştırma Tasarımı	19
Şekil 3.1. Anket katılımcılarının unvan dağılımları.....	31
Şekil 3.2. Anket katılımcılarının sektörel dağılımları.....	32
Şekil 3.3. Kullanılan yazılım yaşam döngüsü modeli.....	33
Şekil 3.4. KOD Yazılım Geliştirme v.1 yüksek seviye gösterimi	47
Şekil 3.5. Analiz ve tasarım fazı	48
Şekil 4.1. KOD Yazılım Geliştirme Modeli Son Sürüm Üst Seviye Gösterimi	57
Şekil 4.2. Yineleme 0 aktiviteleri.....	62
Şekil 4.3. Analiz ve tasarım	63
Şekil 4.4. Değerlendirme fazı.....	67
Şekil 4.5. Değerlendirme kriterlerinden bazıları.....	68
Şekil 4.6. Döngülerde geliştirme adımlarına harcanan zaman.....	71
Şekil 4.7. Müşteri ve kullanıcı rolleri	73

TABLolar DİZİNİ

Tablo 1.1. Kullanılabilirliğin uygulanmasına yönelik daha önce gerçekleştirilen çalışmalar	10
Tablo 2.1. Anket katılımcılarının demografik bilgileri	21
Tablo 2.2. Görüşme katılımcılarının demografik bilgileri	22
Tablo 2.3. Model geçerleme görüşmesine katılanların demografik bilgileri	23
Tablo 2.4. Uygulanacak adımlar ve zaman çizelgesi	24
Tablo 2.5. Görüşme analizi kategorileri	27
Tablo 2.6. Kullanıcı deneyimi tasarım dersi projesi öğrenci yorumları	27
Tablo 3.1. İBE/Kullanılabilirlik kavramının organizasyonlarda anlaşılması ve uygulanması	33
Tablo 3.2. Kullanılabilirlik yöntem ve tekniklerinin uygulandığı yazılım geliştirme aşamaları	34
Tablo 3.3. Kullanılabilirliğin uygulanamamasının sebepleri (çoklu seçim)	35
Tablo 3.4. Kullanılabilirlik değerlendirme yöntemlerini kullanma sıklığı	36
Tablo 3.5. Yazılım geliştirme profesyonellerinin kullanılabilirlik ile ilgili algıları	37
Tablo 4.1. KOD senaryo formatı örneği	58
Tablo 4.2. DOG (BDD)'nin KOD senaryoları kullanımı ile KOD Yazılım Geliştirme Modeli'nde kullanım yeri ve amaçları	58
Tablo 4.3. Modelde müşteriye koruma altına alan durum incelemeleri	59
Tablo 4.4. Modelde yazılımcıyı koruma altına alan durum incelemeleri	60
Tablo 4.5. Yineleme plan formunun bir kısmı	62
Tablo 4.6. Yüksek uygunluk ve düşük uygunluk karşılaştırması	64
Tablo 4.7. Analiz ve tasarım	65
Tablo 4.8. Gerçekleştirme aşaması	66
Tablo 4.9. Karar tablosu	69
Tablo 4.10. Değerlendirme aşaması	70
Tablo 4.11. Yineleme X için örnek yineleme plan formu	74

SİMGELER VE KISALTMALAR DİZİNİ

Kısaltmalar

API	: Application Programming Interface (Uygulama Programlama Arayüzü)
ASD	: Adaptive Software Development (Uyarlanabilir Yazılım Geliştirme)
ATDD	: Acceptance Test Driven Development (Kabul Testi Odaklı Geliştirme)
BDD	: Behaviour Driven Development (Davranış Odaklı Geliştirme)
BÖTE	: Bilgisayar ve Öğretim Teknolojileri Eğitimi
BT	: Bilişim Teknolojisi
DDD	: Domain Driven Development (Alan Odaklı Geliştirme)
DOG	: Davranış Odaklı Geliştirme
ISO	: International Organization for Standardization (Uluslararası Standartlar Organizasyonu)
İBE	: İnsan Bilgisayar Etkileşimi
KOD	: Kullanıcı Odaklı Davranışsal
KTOG	: Kabul Testi Odaklı Geliştirme
RAD	: Rapid Application Development (Hızlı Uygulama Geliştirme)
TDD	: Test Driven Development (Test Odaklı Geliştirme)
TOG	: Test Odaklı Geliştirme
UAT	: User Acceptance Testing (Kullanıcı Kabul Testi)
UCB	: User Centered Behavioral (Kullanıcı Odaklı Davranışsal)
UCD	: User Centered Design (Kullanıcı Odaklı Tasarım)
UML	: Unified Modelling Language (Tümleşik Modelleme Dili)
UX	: User Experience (Kullanıcı Deneyimi)
UXD	: User Experience Desing (Kullanıcı Deneyimi Tasarımı)
UYG	: Uyarlanabilir Yazılım Geliştirme
YM	: Yazılım Mühendisliği
YPF	: Yineleme Plan Formu

KULLANICI ODAKLI DAVRANIŞ GÜDÜMLÜ YAZILIM GELİŞTİRME MODELİ

ÖZET

Günümüzde geliştirilen yazılım projelerinde kullanılabilirliğin ne kadar sağlanabildiği önemli bir konudur. Başarılı bir yazılım geliştirme döngüsü sadece yazılım mühendisliği alanındaki uygulamaların yer almasıyla mümkün değildir. İnsan Bilgisayar Etkileşimi alanından doğmuş kullanılabilirlik kavramının yazılım ile entegrasyonu sağlanamadığı müddetçe o sistem başarılı kabul edilememektedir. Gerçekleştirilen tez çalışmasında bu probleme çözüm bulmak için kullanıcı merkezli ve davranış güdümlü bir yazılım geliştirme modeli olan Kullanıcı Odaklı Davranışsal (KOD) Yazılım Geliştirme Modeli önerilmektedir.

Çalışma kapsamında kullanılabilirlik ve kullanıcı merkezli yazılım geliştirme ile ilgili organizasyonlardaki uygulamaların durumunu anlamak ve sonuçlar çıkarabilmek için karma yöntem yaklaşımlarından çeşitleme araştırma metodolojisi uygulanmıştır. Bu doğrultuda bir anket çalışması yapılmış daha sonrasında ise yine ilgili organizasyonlarda çeşitli pozisyonlarda çalışan kişiler ile birebir görüşmeler gerçekleştirilerek durumun detaylı analizleri gerçekleştirilmiştir. Anket ve görüşme bulguları modelin gelişimine katkı sağlamıştır.

KOD Yazılım Geliştirme Modeli ile kullanılabilir yazılımlar geliştirmeyi sağlayacak şekilde yazılım mühendisliği ve İBE alanlarından çeşitli araç ve teknikleri entegre ederek kullanıcı merkezli bir yazılım geliştirme süreci önerilmektedir. Temel olarak model, yazılım mühendisliğindeki çevik yaklaşımlardan gereksinimlerin daha iyi şekilde analiz edilmesini sağlayan DOG (Davranış Odaklı Geliştirme) ve İnsan Bilgisayar Etkileşimi alanından stil kılavuzları, kullanılabilirlik testleri, sezgisel değerlendirme, Wizard Of Oz teknikleri gibi iyi pratiklerin entegre edildiği yinelemeli bir yaklaşımdır.

Anahtar Kelimeler: Çevik Metodoloji, Davranış Odaklı Geliştirme (DOG), İnsan Bilgisayar Etkileşimi, Kullanıcı Odaklı Tasarım, Kullanıcı Odaklı Yazılım Geliştirme Modeli.

USER CENTERED BEHAVIOR DRIVEN SOFTWARE DEVELOPMENT MODEL

ABSTRACT

Nowadays, it is very important for projects to what extent usability is applied. A successful development life cycle cannot be achieved only with software engineering practices. A development life cycle cannot be considered successful unless integration of usability, a concept introduced by Human Computer Interaction (HCI) field, to software engineering practices is not achieved. In this study, in order to address this problem, User Centered Behavioral (UCB) Software Development Model is proposed.

In this study, in order to understand the current situation in the software development companies in terms of usability and user centered software development triangulation methodology of mixed methods is used. A survey is conducted to identify the current status of the software development companies. Besides interviews are conducted with the employees of these companies to make a detailed analysis of the situation. The survey and the interviews have contributed to the development of the proposed model.

UCB Software Development Model proposes a software development process in which several tools and techniques from software engineering and HCI are integrated in order to create usable software. In essence, UCB Software Development Model proposes an iterative process that combines Behavior Driven Development (BDD), an agile approach that enables a better understanding and analysis of the requirements, with style guides, usability tests, heuristic evaluation and Wizard of Oz techniques of HCI.

Keywords: Agile Methodology, Behavior Driven Development (BDD), Human Computer Interaction, User Centered Design, User Centered Software Development Model.

GİRİŞ

Günümüzde yazılım geliştiren organizasyonlar kullanılabilir yazılım sistemleri geliştirmeyi hedeflemekte ve yazılım geliştirme süreçlerine bunu sağlayacak teknikleri ve aktiviteleri entegre etmeye çalışmaktadırlar (Ferreira ve diğ., 2007). İnsan Bilgisayar Etkileşimi (İBE) alanındaki ana konulardan biri olan kullanılabilirlik kavramı yazılım geliştirme dünyasında da kalite kriterlerinden biri olarak kabul görmesine rağmen, yazılım geliştirme süreçlerinde kullanılabilirliğin sağlanmasına yönelik olarak açık ve tanımlı aktiviteler yer almamaktadır. İBE ve yazılım mühendisliği alanları birbirine yakın disiplinler olmasına rağmen iki alanın ortalama bir yazılım geliştiricinin anlayacağı biçimde ortak kavramlar oluşturularak birbirine entegre edilmesi gerekliliği bulunmaktadır (Ferreira ve diğ., 2007).

İBE ile ilgili konular genellikle yazılım geliştirme yaşam döngüsünün son adımlarında ve daha çok kullanıcı ara yüzü seviyesinde düşünülmektedir (Zhang ve diğ., 2005). Ancak bu konuların daha erken aşamalarda uygulanmasının gerekliliği yazılım geliştiren organizasyonlar tarafından da kabul edilmeye başlanmıştır. Yazılım mühendisliği ve İBE alanlarının entegrasyonu ile ilgili konular pek çok çalışmada karşımıza çıkmaktadır. Bu entegrasyon çabaları geleneksel yöntemlerden (Mayhew, 1999; Zhang ve diğ., 2005; Anderson ve diğ., 2001) daha güncel çevik yöntemlere (Mammel, 2007; Lee ve diğ., 2009; Hellmann, 2010; Losada ve diğ., 2012) geniş bir spektrumda görülmektedir.

Geleneksel yöntemlere göre çevik yöntemlerle İBE konularının adaptasyonu günümüzde daha çok ön plana çıkmaktadır (Mammel, 2007). Yine hem İBE hem de yazılım mühendisliğinde de kullanılan prototipleme yaklaşımının kullanımı paydaşlar arasındaki iletişim için araç olarak görülmektedir (Lee ve diğ., 2009).

Kullanıcı gereksinimlerini ortaya koymak için kullanıcı senaryolarının kullanımı uygulanmaktadır (Losada ve diğ., 2012). Bunlara ek olarak da kişilik kartlarının (persona) kullanımı tasarımın temelini oluşturma ve iletişim için bir araç sağlamada

etkilidir (Hartson ve Hix, 1989). Teste dayalı yaklaşımların da faydalı olduğu görülmektedir. Bu yaklaşımlar herhangi bir fonksiyonel kod yazmadan önce test senaryolarının oluşturulmasına dayalıdır (Lee ve diğ., 2009). Teste dayalı geliştirmeden türetilen davranış odaklı geliştirme yaklaşımında (URL-8, 2006) ise test kelimesinin yeri davranış ile değiştirilmiş ve odak noktası sistemin teknik detaylarından çok, davranışı üzerine yönelmiştir. Davranış Odaklı Geliştirme (DOG - Behaviour Driven Development-BDD) yaklaşımı ile projede yer alan paydaşlardan müşteri, kullanıcı, yazılımcı, analist ve test ekiplerinin ortak dilde konuşabilmesi sağlanmaktadır (Wynne ve Hellesoy, 2012).

Bu tez çalışması kapsamında kullanılabilir yazılımlar geliştirmeyi sağlayacak şekilde yazılım mühendisliği ve İBE alanlarından araç ve teknikler entegre edilerek kullanıcı merkezli bir yazılım geliştirme süreci önerilmektedir. Önerilen yöntemde temel olarak kullanıcı ya da müşterilerin kendisini doğrudan anlatabilmesine olanak sağlayarak gereksinimlerin daha iyi şekilde anlaşılmasını sağlama yönünde sahip olduğu potansiyel doğrultusunda DOG yaklaşımı adapte edilmiş ve bu nedenle model Kullanıcı Odaklı Davranışsal (KOD) Yazılım Geliştirme modeli olarak adlandırılmıştır. Bu model ile gereksinimlerin doğru anlaşıldığı ve geliştirme sürecinin erken aşamalarında kullanıcıyı merkezde tutan yinelemeli çevik bir yöntem önerilmeye çalışılmaktadır. Önerilen modelin erken aşamalarında DOG senaryolarına benzer şekilde tanımlanan KOD senaryoları tanımlanmakta ve bunlara dayalı olarak geliştirilen prototipler kullanılarak kullanıcıların geliştirme sürecine dâhil edilmeleri sağlanmaktadır. Yine ilk aşamalarda geliştirilen uygulamaya yönelik olarak stil kılavuzları tanımlanarak her yinelemede birbiriyle tutarlı tasarımların geliştirilmesi hedeflenmektedir. Modelin gerçekleştirmeye yönelik olan aşamalarında otomatize edilmiş testlerle kullanıcının davranışları simüle edilebilmekte ve böylece geliştirilen yazılımın birçok kez kullanıcı testlerinden geçirilmesi sağlanmaktadır. Böylece kullanıcılar tarafından sistemin yanlış anlaşılma riski minimuma indirilerek hata yapma olasılığının azaltılması planlanmaktadır.

Tez çalışması kapsamında kullanılabilir yazılım geliştirme sağlayacak modeli önerebilmek için aşağıdaki araştırma sorularına yanıtlar aranmaktadır;

- Kullanılabilir yazılım geliřtirmenin saęlanabilmesi için İBE ve yazılım mühendislięi alanından entegre edilmesi gereken önemli iyi pratikler nelerdir?
- Kullanılabilir yazılım geliştirme için hangi aktiviteler gerçekleştirilmelidir?

Bu tez çalışması beř bölümünden oluşmaktadır. Öncelikle bu ilk giriş bölümünde tezin konusu ve amacı açıklanarak çalışma kapsamında yanıtları aranan araştırma soruları verilmiş ve tezin bölümlerinin içerikleri özetlenmektedir. Tezin bundan sonraki birinci bölümünde önerilen modelin oluşturulması ile ilişkili olan İBE ve yazılım mühendislięi alanlarından bu modelde adapte edilen araç ve teknikler ile ilgili temel bilgiler verilmektedir. Daha önce gerçekleştirilmiş olan benzer model önerme çalışmaları özetlenerek bu tezde önerilen modelden farkları ortaya konulmakta ve böylece önerilen modelin anlamlılıęı kanıtlanmaktadır. İkinci bölümde tez çalışmasında kullanılan araştırma yöntemi detaylı olarak açıklanmaktadır. Bu bölümde önce araştırma stratejisi, uygulanan stratejinin nedenleri ve araştırma tasarımı açıklanmakta, daha sonra kullanılan veri toplama teknikleri ve bulguların nasıl analiz edildięi sunulmakta ve son olarak da çalışmanın güvenilirlięi tartışılmaktadır. İkinci bölümde çalışma kapsamında anket ve görüşmeler yolu ile elde edilen bulguların analizleri sunulmaktadır. Yine bu bölüm içerisinde önerilen modelin birinci sürümü de açıklanmaktadır. Üçüncü bölümde önerilen KOD yazılım geliştirme modeli, modelin tüm fazları, modelde yer alacak paydařların rol ve sorumlulukları ile birlikte çeřitli örnekler üzerinden detaylı olarak sunulmaktadır. Son olarak dördüncü bölümde çalışmanın ana sonuçları, güçlü ve eksik yönleri ile gelecekte yapılacak çalışmalar tartışılmaktadır.

1. LİTERATÜR TARAMASI

Bu bölümde bu tez çalışması kapsamında önerilen modelin geliştirilmesinde kullanılan alanlar ve temel kavramlar hakkında temel bilgiler verilmektedir. İlgili alanlardan adapte edilen prensipler, araç ve teknikler listelenerek açıklanmaktadır. Bunlara ek olarak daha önce yazılım mühendisliği ve İBE alanlarını entegre ederek gerçekleştirilmiş model önerme çalışmaları sunulmaktadır.

1.1. Yazılım Mühendisliği ve Yazılım Geliştirme Modelleri

Yazılım mühendisliği alanında kaliteli sistemler geliştirmek için görevleri etkin ve verimli şekilde gerçekleştirecek çeşitli modeller önerilmiştir. Bu sistemler geliştirilirken yazılım mühendisliğinin lineer süreçlerinden etkilendiği görülmüştür. Yazılım mühendisliğinde ana lineer model yaklaşım Şelale Modeli'dir.

Şelale modeli yazılım sürecinin zorlukları göz önüne alındığında yinelemeli olmadığı için eleştirilmektedir. Bu modelde adımlar sırayla takip edilir ve bir adım bitmeden bir sonraki adıma geçilememektedir. Hataların düzeltilebilmesine olanak sağlamamaktadır. Yaşam döngüsü çok uzun olabilmekte ve bu nedenle başlangıçta belirlenen gereksinimlerin güncelliğini yitirme riski ortaya çıkmaktadır. Kullanıcılar çalışan sistemi sadece yaşam döngüsünün son adımında görebilmektedir (Sommerville, 2004).

Şelale modeli ya da yinelemeli olmayan modellerin dezavantajlarını ortadan kaldırmak için bazı evrimsel modeller önerilmiştir. 80'lerin ortalarında prototipleme yaklaşımı ortaya konulmuştur.

Erken yinelemeli geliştirme modeli olan Spiral Model Boehm (1986) tarafından önerilmiştir. Spiral model hem tasarım hem de prototipleme unsurlarını birleştirmiştir. 90'lı yılların sonunda yeni bir yazılım geliştirme yaklaşımı olan çevik geliştirme metodolojisi ortaya atılmıştır. Çevik ve yinelemeli yöntemler lineer modellere alternatif olan yaklaşımlardandır. En iyi uygulamalar üzerine temellendirilmiş bu yaklaşımlar ticari olarak da başarılarını kanıtlamışlardır.

2001’de yayınlanan agile manifesto çevik yaklaşım kurallarını içermektedir. Çevik yöntemlerin temel özellikleri agile manifestoda şu şekilde belirtilmiştir (URL-10, 2001):

1. En önemli önceliğimiz değerli yazılımın erken ve devamlı teslimini sağlayarak müşterileri memnun etmektir.
2. Değişen gereksinimler yazılım sürecinin son aşamalarında bile kabul edilmelidir. Çevik süreçler değişimi müşterinin rekabet avantajı için kullanır.
3. Çalışan yazılım, tercihen kısa zaman aralıkları belirlenerek birkaç haftada ya da birkaç ayda bir düzenli olarak müşteriye sunulmalıdır.
4. İş süreçlerinin sahipleri ve yazılımcılar proje boyunca her gün birlikte çalışmalıdırlar.
5. Projelerin temelinde motive olmuş bireyler yer almalıdır. Onlara ihtiyaçları olan ortam ve destek sağlanmalı, işi başaracakları konusunda güven duyulmalıdır.
6. Bir yazılım takımında bilgi alışverişinin en verimli ve etkin yöntemi yüz yüze iletişimidir.
7. Çalışan yazılım ilerlemenin birincil ölçüsüdür.
8. Çevik süreçler sürdürülebilir geliştirmeyi teşvik etmektedir. Sponsorlar, yazılımcılar ve kullanıcılar sabit tempoyu sürekli devam ettirebilmelidir.
9. Teknik mükemmeliyet ve iyi tasarım konusundaki sürekli özen çevikliği artırır.
10. Sadelik, yapılmasına gerek olmayan işlerin mümkün olduğunca arttırılması sanatı, olmazsa olmazlardandır.
11. En iyi mimariler, gereksinimler ve tasarımlar kendi kendini örgütleyen takımlardan ortaya çıkar.
12. Takım, düzenli aralıklarla nasıl daha etkili ve verimli olabileceğinin üzerinde düşünür ve davranışlarını buna göre ayarlar ve düzenler.

Çevik ve yinelemeli yöntemlerin temel alındığı en iyi uygulamalar şu şekilde sıralanabilmektedir (Kruchten, 1999).

- Yazılımın yinelemeli olarak geliştirilmesi,
- Gereksinimlerin yönetilmesi,
- Bileşen tabanlı mimarilerin kullanımı,
- Yazılımın görsel olarak modellenmesi
- Yazılımın kalitesinin kontrolü,

- Yazılımdaki değişikliklerin kontrol altında tutulması

Çevik yöntemler arasında yer alan Uyarlanabilir Yazılım Geliştirme ise proje yönetimi ve işbirliğine daha fazla vurgu yapan yaklaşımlardan biridir (Highsmith, 2002). Bu yaklaşımın detayları 1.1.1 başlığı altında anlatılmaktadır.

1.1.1. Uyarlanabilir yazılım geliştirme (ASD)

Çevik yaklaşımlardan olan Uyarlanabilir Yazılım Geliştirme (Adaptive Software Development-ASD) hızlı değişen projelerin performansını arttırmayı amaçlayan Hızlı Uygulama Geliştirme'den (Rapid Application Development-RAD) doğmuş yinelemeli bir yaklaşımdır. UYG Yaşam Döngüsünün elemanları:

- Üzerinde düşünmek(Speculate),
- İşbirliği yapmak,
- Öğrenmektir.

UYG yaklaşımında zaman kısıtlı yaklaşım ile döngülerin zamanları planlanmaktadır. Bu tekniğe göre diğer çevik yaklaşımlara göre daha esnek zamanlama imkânı sunulmaktadır. Döngülerin periyotları UYG modelinde, 9 aydan az projeler için 4-8 hafta, 9 aydan fazla zaman alacak projeler için 6-10 hafta arasında değişebilmektedir. Projelerin karmaşıklık seviyesine göre her döngünün periyodik zamanı farklılık gösterebilmektedir (Highsmith, 2002).

1.1.2. Davranış odaklı geliştirme (BDD)

Çevik yaklaşımlardan biri olan Davranış Odaklı Geliştirme son yıllarda yazılım projelerinde sık kullanılmaya başlamıştır. DOG'u geleneksel yöntemlerden ayıran en önemli özelliklerden biri geleneksel kullanıcı senaryolarında yalnızca fonksiyonalliteye odaklanılırken, DOG senaryolarında fonksiyonallitenin yanı sıra kullanıcı davranışlarına da yer verilmesidir. Bunun yanı sıra DOG senaryolarının yazımında iş birimlerinin de dahil olması sebebiyle senaryolar yazılımcılar, analistler ve müşteriler arasındaki iletişimi kolaylaştırmakta ve teknik detaylardan daha çok sistem davranışı hakkında bilgi vermektedir (Wynne ve Hellesoy, 2012).

DOG(BDD), sistemin davranışlarının yazılım sürecinde yer alan paydaşlarca anlatılmasına odaklanan çevik geliştirme yöntemlerinden biridir. Dan North

tarafından geliştirilen bu dil, son zamanlarda akademik ve endüstriyel alanda da dikkatleri üzerine toplamaktadır (Solis ve Wang, 2011). Test Odaklı Geliştirme (TOG) (Test Driven Development-TDD), Kabul Testi Odaklı Geliştirme (KTOG) (Acceptance Test Driven Development-ATDD), Alan Odaklı Geliştirme (Domain Driven Development-DDD) üzerine temellendirilen DOG, diğer çevik yaklaşımlarla karşılaştırıldığında, paydaşlar için anlaşılabilir bir dille gereksinimleri ifade etmekte ve onlarla sağlıklı iletişim kurmayı vaat etmektedir (Solís ve Wang, 2011). Çünkü gereksinimler konuşma diliyle ifade edildiğinden üzerine tartışmak, paydaşlarca hakkında fikir beyan etmek kolaylaşmaktadır. KTOG (Beck 2003) de TOG (Beck 2003; Astels 2003) gibi uygulamadan önce yazılan testleri içerir. Kabul için kullanılan bu testler yazılımda gerekli olan bir davranışa karşılık gelmektedir. KTOG kabul testleri sırasında ortaya çıkan gereksinimleri tespit edip, bunları geliştirme sürecini yönlendirmeye odaklanır. Bu iki yaklaşım kalite ve verimliliği arttırmak için kullanılmaktadır. DOG müşteri odaklıyken, KTOG daha çok yazılımcı odaklıdır (URL-4, 2014). Sadece yazılım ekiplerince kullanılan ve kullanılırken neyi test edip neyi test etmemeleri gerektiği, nereden başlanması gerektiği, testlerin nasıl adlandırılması ve ne kadar test yazılacağı, testin başarısız olma nedenini nasıl anlamaları gerektiği konusunda yaşanan kafa karışıklıkları DOG ile giderilmektedir. Test” kelimesi “davranış” ile yer değiştirilmiş ve sadece yazılımcıların değil, sistemde yer alan tüm paydaşların anlayabileceği bir formatla yazılması ile DOG’un bu sorunların üstesinden gelmesi sağlanmıştır (URL-8, 2006).

Bir projenin geliştirilmesi sırasında ilk olarak gereksinim analizinin sağlıklı bir şekilde yürütülmesi gerekmektedir. Genellikle projelerin gereksinim analizleri sırasında teknik takım ile yönetim tarafı arasındaki (business) iletişim problemleri, alınan kararlar arasında uyumsuzluklar bu süreci olumsuz etkilemektedir. DOG bu iki tarafın işbirliği yapmasını sağlayarak, ortak paydada buluşturarak, öncelikli olarak ihtiyaç analizinin başarılı yapılmasını sağlamaktadır. Sadece gereksinim analizi aşamasında değil, yazılımın geliştirilmesi safhasında da doğru yazılımı yapılandırmak, yazılımı doğru yapılandırmak, API ve kod tasarımına yardımcı olmak konusunda da çok kullanılan bir yaklaşım olarak kabul görmektedir. Sistemin davranış öğelerini kapsayacak kurallar kümesine kabul kriteri denilmektedir. DOG ile kabul kriterleri Specflow, JBehave, Cucember gibi bazı araçlar yardımıyla

otomatize edilebilmektedir (Smart, 2014). Şekil 3.4 DOG yaklaşımının (Smart, 2014) kullanıldıkları sistemde hangi avantajları sağlayacağını göstermektedir.



Şekil 1.1. DOG Yaklaşımı (Smart, 2014)

1.2. İnsan Bilgisayar Etkileşimi ve Kullanılabilirlik

Arayüz, kullanıcının yazılım ile ilk etkileşime geçeceği nokta olduğundan, kullanıcılar sistemi arayüzü baz alarak değerlendirmektedirler. Bu nedenle İnsan Bilgisayar Etkileşimi konularının geliştirme yaklaşımlarıyla entegre edilmesi kullanıcıların sistemi etkin bir şekilde öğrenmeleri üzerinde önemli rol oynamaktadır (Faulkner, 1998).

Kullanılabilirlik kavramı İBE'nin ergonomi dalından ortaya çıkmış ve İBE'nin teorik yanlarından çok pratik konularına işaret etmektedir (Faulkner, 1998). Kullanılabilirlik "kullanıcının görevini etkin, verimli, kullanıcı memnuniyetini sağlayan bir şekilde gerçekleştirebilmesine imkân tanımak olarak tanımlanmaktadır" (Rosson ve Carroll, 2002). Bu tanımdan anlaşılacağı üzere kullanılabilirlik temel olarak "öğrenilebilirlik" "etkin kullanım" ve "kullanıcı memnuniyeti" konularına odaklanmaktadır.

1.3. Kullanılabilirliğin Yazılım Geliştirme Organizasyonlarında Uygulanması

Günümüzde geliştirilen yazılım projelerinde kullanılabilirlik önemi kazanarak, başarı kriterlerinden biri olarak sayılmaya başlamıştır. Geliştirilen projelerin kullanıcı merkezli olabilmesi için kullanılabilirlik kavramının iyi anlaşılması ve doğru uygulanması gerekmektedir. Kullanılabilirlik, geliştirilen bir sistemin insanlar

tarafından kolay öğrenilmesini, insanların işlerini kolaylaştıran fonksiyonları içermesini, kullanımının kolay ve memnuniyet verici olmasını sağlayan İnsan Bilgisayar Etkileşimi (İBE) alanından doğmuş bir kavramdır (Gould ve Lewis, 1985). Aynı zamanda yazılım geliştirme perspektifinden de kullanıcı performansı, memnuniyet ve öğrenilebilirliğe karşılık gelen bir yazılım kalite niteliğidir (Seffah, 2004). Kullanılabilirlik kavramı ve yazılım geliştirme, geliştirilecek ürünlerin başarısı açısından birbirleriyle yakından ilişkilidir. Kullanılabilirliğin yazılım yaşam döngüsüne entegrasyonuna yönelik pek çok çalışma yer almaktadır (Seffah ve Metzker, 2004; Göransson ve diğ.,2003)

Kullanılabilirlik kavramı hem İBE hem de yazılım mühendisliği alanının kesişimini sağlayan bir kavram olmasına karşın her iki alan tarafından farklı algılanmaktadır. Yazılım geliştirme daha çok sistemin fonksiyonları ile ilişkiliyken, kullanılabilirlik daha çok kullanıcıya odaklanır. Her iki alanda da kullanıcı oldukça önemlidir ancak rolü konusunda farklılık vardır. Yazılım geliştirmede kullanıcı gereksinimlerin birincil kaynağı olarak görülürken, kullanılabilirlik tarafında ise sistemi tasarlamamanın birinci kaynağıdır (Bygstad ve diğ., 2008)

Kullanılabilirlik BT alanında da geniş kabul gören bir kavram olmasına karşın (Gulliksen ve diğ., 2004) yazılım geliştirme süreçlerinde uygulanması beklenildiği kadar yaygın değildir. Bunun en önemli nedeni de BT organizasyonlarının kullanılabilirliğin karmaşık olduğuna ve uygulanmasının fazla kaynak (zaman ve maliyet) gerektirdiğine yönelik inancıdır (Ji ve Yun, 2006).Bu düşük uygulanma ve yanlış anlaşılmanın giderilmesi için güncel durumun ne olduğunun araştırılması önemli bir konudur. Kullanılabilirliğin özellikle etkileşimli sistemler başta olmak üzere tüm yazılımlar için önemli bir özellik olduğu genel olarak kabul görmüş olsa da (Bygstad ve diğ., 2008), pratikte yazılım geliştiren organizasyonlarda ne kadar yaygın ve ne şekilde uygulandığı pek çok araştırmacının ilgilendiği bir konu olmuştur.

Buna yönelik olarak son 30 yıl içerisinde çeşitli ülkelerde ulusal ya da uluslararası ölçekte anket çalışmaları düzenlenmiş ya da görüşme yoluya daha detaylı tespitler yapılmaya çalışılmıştır. Çalışmalar Tablo 1.1’de özetlenmektedirler.

Tablo 1.1. Kullanılabilirliğin uygulanmasına yönelik daha önce gerçekleştirilen çalışmalar

Araştırma Referansı	Araştırmanın uygulandığı bölge	Yıl	Katılımcı sayısı	Araştırma konusu / alanı
(Gould ve Lewis, 1985)	Raporlanmadı	1985	447	Kullanılabilirlik prensiplerine yönelik tasarımcıların algısı
(Dillo ve diğ., 1993)	9 Avrupa ülkesi	1993	84	Kullanıcı merkezli tasarım, kullanılabilirlik kavramlarına yönelik algı ve endüstrideki pratikler
(Rosenbaum ve diğ., 2000)	CHI ve UPA Konferansı katılımcıları	2000	Raporlanmadı	Kullanılabilirliğin önündeki engeller ve uygulanan yöntemler
(URL-6, 2001)	Raporlanmadı	2000	100	Kullanılabilirliğin önündeki engeller ve uygulanan yöntemler
(Vredenburg ve diğ., 2002)	USA ve Europe	2002	102	Kullanılabilirliğin uygulanması ve kullanılan yöntemler
(Gulliksen ve diğ., 2004)	İsveç	2003	273	Kullanılabilirliğin endüstrideki durumu, kullanılan yöntem ve teknikleri
(Venturi ve Troost, 2004)	Raporlanmadı	2004	83	Kullanıcı merkezli tasarım ve kullanılabilirlik yöntemleri
(Ji ve Yun, 2006)	Kore	2006	274	Kullanılabilirlik pratikleri ve problemleri
(Zhou ve diğ., 2008)	Çin	2008	404S	Kullanıcı merkezli tasarım ve kullanılabilirlik yöntemleri
(Bygstad ve diğ., 2008)	Norveç	2008	78 şirket	Yazılım geliştirme yöntemleri ve kullanılabilirliğin durumu
(Bak ve diğ., 2008)	Danimarka	2008	39 yazılım geliştirici organizasyon	Kullanılabilirliğin önündeki engeller

Tablo 1.1. (Devam) Kullanılabilirliğin uygulanmasına yönelik daha önce gerçekleştirilen çalışmalar

(Hussain ve diğ.,2009)	Birçok ülke	2009	92	Çevik yöntemler ve kullanılabilirliğin entegrasyonu
(Ardito ve diğ., 2011)	İtalya	2011	Raporlanmadı	Kullanılabilirliğin önündeki engeller (Bak ve arkadaşlarının (2008) çalışmasının tekrarı)
(Darnell ve Halgren, 2008)	Birçok ülke	2009	34 ülke	Kullanılabilirliğin endüstrideki durumu
(Chilana ve diğ., 2011)	Kuzey Amerika, Asya ve Avrupa	2011	333 katılımcı	Kullanılabilirliğin teslimat sonrası aşamalarda kullanımı
(Birant ve Boz, 2011)	Türkiye	2011	50 katılımcı	Kullanılabilirlik analizlerinin kullanımı

Kullanılabilirlik ve kullanıcı merkezli tasarıma yönelik organizasyonlarda genel olarak bir farkındalık bulunmaktadır (Gould ve Lewis, 1985; Ji ve Yun, 2006; Dillon ve diğ., 1993). Kullanılabilirliğin uygulanması % 42 (Ji ve Yun, 2006) gibi oranlarla oldukça düşük olarak gözlemlenmektedir (Venturi ve Troost, 2004; Zhou ve diğ., 2008). Bunun tersi yönünde bulguları olan çalışmalar da mevcuttur ancak bu çalışmalarda da kullanılabilirlik kavramının daha çok fonksiyonellik kavramı ile karıştırılması nedeniyle bu yönde bir sonuç çıktığı görülmektedir (Hussain ve diğ., 2009).

Bu farkındalık ve kavramın daha doğru anlaşılması konusunda yıllar geçtikçe iyileşme olduğu görülmekte ve kullanıcı merkezli tasarımın ve kullanılabilirliğin daha yaygın olarak kullanıldığı tespit edilmektedir (Vredenburg ve diğ., 2002; Darnell ve Halgren, 2008).

Kullanılabilirliği yaygın kullanımın önündeki engeller olarak temel kavramların doğru anlaşılabilmesi (Gould ve Lewis, 1985; Dillon ve diğ., 1993; Rosenbaum ve diğ., 2000; URL-6, 2001; Hussain ve diğ., 2009), geliştiricilerin konuya karşı dirençli olmaları (Rosenbaum ve diğ., 2000; URL-6, 2001; Hussain ve diğ., 2009), yönetimin yeterince desteğinin olmayışı (Venturi ve Troost, 2004), zaman ve kaynak kısıtları (URL-6, 2001; Hussain ve diğ., 2009), altyapı ve iletişim konularındaki eksiklikler (Venturi ve Troost, 2004), eğitilmiş personel olmayışı (Rosenbaum ve diğ., 2000),

testlerin yapılmasındaki zorluklar ve müşteri katılımının sağlanamaması (Hussain ve diğ., 2009) listelenmektedir. Diğer taraftan kullanılabilirliğe karşı artan farkındalık doğrultusunda, geliştiriciler arasında kullanılabilirliğin uygulanmasının müşteri memnuniyetine olumlu etkisinin olduğu (Ji ve Yun, 2006; Vredenburg ve diğ., 2002; Zhou ve diğ., 2008; Ardito ve diğ., 2011), ürün geliştirme maliyet ve zamanını azalttığı (Zhou ve diğ., 2008), ürün kalitesini artırdığı (Ardito ve diğ., 2011) ve proje başarısına olumlu etkisinin olduğu (Bak ve diğ., 2008) yönünde düşünceler yaygınlık kazanmaktadır.

Kullanılabilirliğin sağlanmasına yönelik olarak daha çok resmi olmayan kullanıcı ya da uzman içermeyen testlerin organizasyonlar içerisinde uygulanması yaygındır (Dillon ve diğ., 1993; Vredenburg ve diğ., 2002). Sezgisellere dayalı yöntemlerin etkilerinin az olmaları bilinmesine karşılık düşük maliyetli olmaları nedeniyle tercih edildikleri görülmektedir (Rosenbaum ve diğ., 2000; Vredenburg ve diğ., 2002). Kullanılabilirlik testleri ise etkinlikleri en fazla olan değerlendirme yaklaşımı olarak kabul edilmektedir (Rosenbaum ve diğ., 2000; Birant ve Boz, 2011) ve bazı çalışmalarda sık kullanıldığına dair bulgular elde edilmiştir. Kullanıcıların dahil olduğu sesli düşünme (Göransson ve diğ., 2003), prototipleme (Göransson ve diğ., 2003; Venturi ve Troost, 2004; Birant ve Boz, 2011), görüşme (Göransson ve diğ., 2003; Venturi ve Troost, 2004), kullanıcı araştırmaları (Göransson ve diğ., 2003; Birant ve Boz, 2011), odak grupları (Zhou ve diğ., 2008) en sık tercih edilen değerlendirme yöntemleridir.

Kullanılabilirliğin organizasyonlarda son yıllarda yaygın kullanılmaya başlanmasına dayalı olarak, kullanılabilirlik çalışmalarında yer alan çalışanların ortalama görev süreleri genellikle 7 yıl olarak tespit edilmektedir (Vredenburg ve diğ., 2002; Darnell ve Halgren, 2008). Çalışanların kullanılabilirlikle ilgili bilgileri ise genellikle resmi bir eğitimden değil (Vredenburg ve diğ., 2002) daha çok işte deneyimleyerek (Göransson ve diğ., 2003) kazanıldığı görülmektedir. Organizasyonlarda yaygın olarak kullanılan yaşam döngüsü modeli olarak genellikle organizasyonların kendi dahili süreçlerini takip ettikleri (Göransson ve diğ., 2003), bunun dışında en yaygın kullanılan yöntem olarak çevik yöntemler (Hussain ve diğ., 2009) ve çevik yöntemler arasından da RUP (Bygstad ve diğ., 2008; Hussain ve diğ., 2009), SCRUM (Hussain ve diğ., 2009) ve XP (Hussain ve diğ., 2009) kullanılmakta olduğu ortaya

çıkılmaktadır. Çeşitli çalışmalarda çalışanlar tarafından kullanılabilirliğin daha iyi uygulanabilmesi için kullanılabilirlikle ilgili bir süreç ve metodolojiye ihtiyaç olduğu vurgusu yapılmaktadır (Ji ve Yun, 2006; Dillon ve diğ., 1993). Kullanılabilirliğin değerlendirilmesinin yazılım yaşam döngüsünde doğrudan ve dolaylı kullanımı başlamış olmasına (Birant ve Boz, 2011) rağmen daha çok tasarım aşamasında uygulanırken teslimat sonrası aşamada kullanım oranı oldukça düşük kalmaktadır (Chilana ve diğ., 2011).

Bu çalışmalarda ortak olarak çıkan sonuç kullanılabilirlik ve kullanıcı merkezli tasarım konusunda yazılım organizasyonlarda genel bir farkındalık ve olumlu bir algı olmasına karşılık pratikte uygulanma oranları oldukça düşük kalmaktadır. Bu kavramların yazılım geliştiren organizasyonlar tarafından ülkemizde ne kadar uygulandığına yönelik olarak gerçekleştirilen bu çalışma ile yazılım geliştiren organizasyonlarda kullanılabilirlik değerlendirmelerinin uygulanma durumu ve kullanılabilirlik konusunda yazılım geliştiricilerinin sahip oldukları algının ne olduğu belirlenmeye çalışılmaktadır. Bu çalışmanın sonuçlarının kullanıcı merkezli tasarımın planlanması ve uygulanması için faydalı olacağı öngörülmektedir.

1.4. Kullanılabilirliğin Yazılım Yaşam Döngüsüne Entegrasyonu ile İlişkili Çalışmalar

Günümüzde kullanılabilirliğin öneminin artmasıyla birlikte uygulanan yazılım yaşam döngüsü modelleri tek başına değerlendirilmemektedir. Bu döngüye kullanılabilirlik de entegre edilmeye çalışılmaktadır. Hussain ve diğ. (ISO/IEC,1999) geliştirme döngüsüne entegre edilebilecek farklı İBE enstrümanları önermiştir. Bunlar kullanıcı çalışmaları, “persona”lar, kullanılabilirlik uzman değerlendirmeleri, kullanılabilirlik testleri, geliştirilmiş birim testler ve “lightweight” prototiplerdir. “Persona”ların kullanımı Haikara (Hartson ve Hix, 1989) tarafından da yazılımcı ve müşteriler arasında bir iletişim aracı olarak önerilmiştir.

Mayhew’ e (1999) göre stil kılavuzlar tüm gereksinim analizlerini ve kullanıcı arayüzü tasarım ürünlerini tek bir doküman altında toplamaktadır. Stil kılavuzlarının erken safhalarda geliştirilmesi önerilmektedir. Mayhew, kullanılabilirlik mühendisliği yaşam döngüsündeki ürünlerin dokümante edilmesini önemli kılan sebepleri aşağıdaki gibi listelemektedir:

- Uzun geliştirme döngülerine sahip projelerde iyi dokümanite edilmemesi halinde herhangi bir bilgi ya da tasarım kararı zaman içinde unutulabilmektedir.
- Özellikle karmaşık projelerde çok fazla sayıda gereksinim ve tasarım kararı ortaya çıkar. Tasarımcıların tüm bunları akıllarında tutmaları oldukça zordur.
- Projede büyük bir ekibin çalışması durumunda dokümantasyon önemli bir iletişim aracı olarak işlev görmektedir.
- Bir projede yapılan işlerin çoğu, başka bir projede yeniden kullanılabilir. Bu nedenle dokümantasyon sonraki projelerde harcanacak gücü azaltmak için önem kazanmaktadır.

Birçok yazılım geliştirme modeli, İBE'ye ilişkin konuları ve sorunları sistematik bir şekilde kapsamamaktadır. Bu konular genellikle kullanıcı arayüzü katmanında ya da yazılım yaşam döngüsünün son evrelerinde dikkate alınmaktadır (Zhang ve diğ., 2005). Ancak, yazılım geliştirme şirketleri bu tekniklerin geliştirme sürecinin erken safhalarında entegre edilmesinin önemini farkına varmışlardır (Anderson ve diğ., 2001; Ferreira ve diğ., 2007). YM ve İBE'ye ilişkin konuların dahil edilmesi bazı çalışmalarda ele alınmıştır. Bu çalışmalar, geleneksel modellerden yakın dönemdeki çevik yaklaşıma kadar geniş bir spektrumda görülmektedir. Hartson ve Hix (1993), bir yıldız model önermiştir. Bu model non-lineer yaklaşımı benimseyen ilk model olarak kabul edilebilir, böylelikle gereksinimler, tasarım ve geliştirme yaşam döngüsü boyunca yavaşça gelişebilmektedir. Darnell ve Halgren (2008) kullanılabilirlik testlerinin nerelerde gerçekleştirilebileceğini göstererek tipik geliştirme yaşam döngüsünü karşılaştırmış ve sadece son safhalarda “high-fidelity” kullanılabilirlik testlerinin yapılmasındansa, tüm yaşam döngüsü boyunca uygulanabilecek çeşitli teknikler önermişlerdir. Bu çalışmalar, kullanılabilirlik tekniklerinin geliştirme sürecinin erken safhalarında uygulanmasının önemini vurgulamaktadır (Anderson ve diğ., 2001; Ferreira ve diğ., 2007).

Artırımlı yinelemeli yaklaşımların kullanımı daha önce geliştirilen birçok modelde vurgulanmıştır (Janzen ve Saiedia, 2005; Memmel ve diğ., 2007; Gulliksen ve diğ., 2004).

Mayhew (Mammel ve diğ., 2007), kullanılabilirlik mühendisliği tasklarını “hızlı prototipleme” veya nesne tabanlı yazılım mühendisliği yaklaşımları kullanılan tüm

geliştirme döngüsüne uygulamıştır. Holzinger ve diğ.'in (2005) “uç kullanılabilirlik geliştirme” metodu çevik geliştirme yöntemler ile İBE'nin entegrasyonuna ilk örneklerden biri olarak sayılabilir. Bu çalışmada kısa yinelemelere, sık yeniden planlamalara ve basit tasarıma odaklanmışlardır. Detwiler' in (Detweiler, 2007) üç yinelemeli “user centered design” yaklaşımı da çevik geliştirme yaklaşımı olarak kabul edilebilir. Onun yaklaşımı, gerçek son kullanıcıların geliştirmenin erken safhalarında sürece dahil edilerek, onların geribildirimlerinin alınmasını içermektedir. Chamberlain ve diğ. (2006) YM ve İBE' nin entegrasyonu için 5 temel noktaya odaklanır. Bunlar her iki yöntemin ortak noktaları olarak da değerlendirilebilir. Bu 5 nokta: kullanıcı katılımı, işbirliği ve kültür, prototip oluşturma, proje yaşam döngüsü ve proje yönetimidir. Memmel ve diğ. (2007) XP tabanlı bir geliştirme modeli olan CRUISER'i önermişlerdir. CURISER, senaryo ve prototiplerin kullanımı ve kullanıcılar ile paydaşların yüksek oranda katılımına odaklanmaktadır. Lee, McCrickard ve Stevens (2009)'in de modellerinin temelinde XP yer almaktadır ve süreçlerini senaryo bazlı tasarım ilkeleri ile entegre etmişlerdir. “Uzman Görüşü” ve “Sezgisel Değerlendirme” gibi hafif kullanılabilirlik değerlendirme metodlarını vurgulamışlardır. Aynı zamanda ön tasarımın geliştirme süreci başlamadan yapılmasının önemini de vurgulamışlardır.

Losada ve diğ. (2012a, 2012b) kullanıcıların hedeflerini baz alan InterMod geliştirme modelini önermiştir. Bu modelde aslında geleneksel modellerin temel fazlarına “yıldız” modelinde olduğu gibi prototiplemenin eklenmesi temeline dayanmaktadır. Bu modelde her fazın girdi ve çıktıların saklandığı ortak bir depolama noktası olarak “ara tanımlama tasarımı (intermediate description design-IDD)” yi kullanmışlardır. Aynı zamanda sürekli değerlendirmenin de önemini vurgulamışlardır.

Humayoun ve diğ. (Humayoun ve diğ., 2011) süreç yaşam döngüsü, yineleme ve geliştirme seviyelerinde geçerli olan bir entegrasyon yaklaşımı önermişlerdir. Süreç yaşam döngüsü seviyesinde uygun UCD metodlarının seçimi, yineleme seviyesinde yineleme planlamasıyla tekniklerin entegrasyonu ve geliştirme seviyesinde ise UCD aktivitelerinin otomasyonunu vurgulamışlardır. Test Odaklı Geliştirme yaklaşımı kullanıcı etkileşimli tasarıma da uygulanmıştır (Horn, 2012). TOG yaklaşımı, TOG'un müşteriler ile yazılımcılar arasında etkili bir iletişim aracı olarak

kullanılabilirliğinden ötürü bu modelde kullanılmıştır. Aynı zamanda TOG yazılımındaki hataların azaltılmasına ve böylelikle yazılım kalitesinin artmasına da olanak sağlamaktadır. Kullanıcı hikâyeleri, arayüzlerin “low fidelity” prototiplerini oluşturmak ve sonrasında bunların kullanılabilirlik açısından değerlendirilmesi için kullanılmaktadır. Da Silva (Da Silva ve diğ., 2013), literatürdeki temel uygulamaları özetlemiştir. Geliştirme başlamadan önce iş analizi sırasında bir “sprint 0”ın yapılması, “index card”larının kullanımı ile “little up front design” yapılması, “sketches on whiteboard”, “low fidelity” prototiplerin kullanımı, kullanıcı hikayelerini temel alan prototiplerin mümkün olan en erken safhada kullanımı kağıt prototipler ve “mockup”lar ile kullanıcı testlerinin yapılması, kullanılabilirlik gereksinimlerinin belirlenebilmesi için kullanıcı hikayelerinin kullanılması ve “inspection evaluation” larının yapılması bu çalışmada özetlenen uygulamalar arasındadır.

En iyi uygulamalar, geleneksel modeller yerine çevik yazılım geliştirme yaklaşımının kullanılması olarak özetlenebilir (Mammel ve diğ., 2007). Çevik geliştirme yöntemleri ve kullanılabilirlik mühendisliğinin ortak noktalarından birisi de her ikisinde de kaliteden ödün vermeden kısa zaman aralıklarında geliştirmelerin yapılması gerekliliğidir. Her ikisi de yinelemeli süreçler olarak değerlendirilebilir (Ferreira ve diğ., 2007; Göransson ve diğ., 2007; Gulliksen ve diğ., 2004). Klasik UCD modelleri kullanıcılar, iş akışları ve içerik açısından daha uzun analiz ve dokümantasyon gerektirmekte ve bu mevcut yazılım geliştirme gereksinimleri ile karşılanamamaktaydı. Prototipleme, özellikle tüm paydaşlar arasındaki iletişimde, en etkili araçlardan biri olarak değerlendirilebilir ve yaşam döngüsünün tüm aşamalarında kullanılmalıdır (Lee ve diğ., 2009; Losada ve diğ., 2012). “Low fidelity” prototiplere dayalı değerlendirmeler geliştirmenin erken safhalarında gerçekleştirilirken, sonraki yinelemelerde “high fidelity” versiyonlar ile daha muntazam değerlendirmeler gerçekleştirilmelidir (Chamberlain ve diğ., 2006; Hussain ve diğ., 2009; Mammel ve diğ., 2007). Kullanıcı senaryoları, kullanıcı arayüzü gereksinimlerini ortaya çıkarmak için etkili bir araç olarak kullanılabilir (Losada ve diğ., 2012). UML gerçek kullanıcı arayüzü davranışlarını görselleştiremediği için, kullanıcı senaryoları UML’de tanımlanan case’lerden daha güçlü olarak değerlendirilir. Bunlara ek olarak, “persona”ların kullanımı tasarımın

temelini oluřturmaya yardımcı olmakta ve bir iletiřim aracı olarak da iřlev yapmaktadır (Hartson ve Hix, 1989;ISO/IEC, 1999). Test odaklı yaklařımlar da yardımcı olabilir ancak bunlarla ilgili bazı kısıtlar bulunmaktadır. Örneęin geliřtirme sürecinde GUI sürekli deęiřebilmekte ve bu durum testlerin sürekli olarak güncellenmesini gerektirmektedir.



2. ARAŞTIRMA YÖNTEMİ

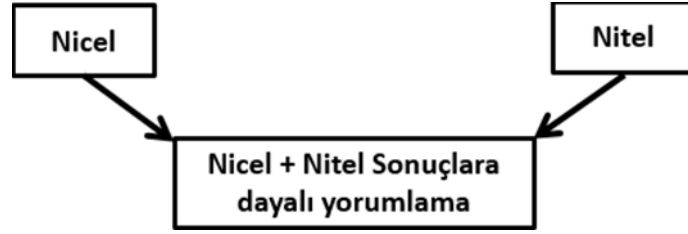
Bu bölümde tez çalışmasında kullanılan araştırma yöntemi açıklanmaktadır. Çalışma kapsamında önerilen modeli geliştirmek için takip edilen adımlar araştırma tasarımı ile sunulmakta, veri toplama araçları, elde edilen verilerin analizinde kullanılan yöntemler ve çalışmanın güvenilirliği tartışılmaktadır.

2.1. Araştırma Stratejisi

Tez çalışması kapsamında kullanılabilirlik ve kullanıcı merkezli yazılım geliştirme ile ilgili organizasyonlardaki uygulamaların durumunu anlamak ve sonuçlar çıkarabilmek için son yıllarda da araştırmacılar tarafından kullanımı oldukça artan karma yöntem (Creswell, 2003) uygulanmıştır. Karma yöntem, araştırma konusunun daha iyi anlaşılabilmesi için tek başına nitel ya da nicel yöntemlerin eğilimleri ve durumun detaylarını yakalamada yetersiz kalabileceği durumlarda, nitel ve nicel verilerin toplanması ve birbirlerini tamamlayarak daha sağlam çözümler yapmayı sağlaması açısından kullanılan bir araştırma yaklaşımıdır. Alanyazında farklı yazarlar tarafından tanımlanmış olan çok sayıda karma yöntem olmasına rağmen (Ivankova ve diğ., 2006; Cameron, 2009), Creswell ve Clark (2007) çeşitleme tasarımı, gömülü tasarım, açıklayıcı tasarım ve keşifçi tasarım şeklinde dört ana tip karma yöntem tanımlamışlardır. Bu tez çalışmasında bu karma yöntemlerden çeşitleme (triangulation) tercih edilmiştir.

Çeşitleme tasarımı karma yöntemler içerisinde en iyi bilenen yaklaşımlardan biridir. Bu tasarımın amacı “aynı konu ile ilgili tamamlayıcı farklı bilgiler elde etmedir” (Creswell ve diğ., 2007).

Şekil 2.1’de görüldüğü üzere çeşitleme tasarımında nitel ve nicel veriler aynı zaman aralığında paralel olarak toplanır ve eşit ağırlıklandırılır ve bu nedenle de eşzamanlı çeşitleme tasarımı adını alır. Ancak bu tez çalışmasında nitel ve nicel verilerin toplanması aynı zamanlarda gerçekleştirilememiş, farklı zaman aralıklarında toplanmış ancak sonuçlar beraber değerlendirilerek yorumlanmaya çalışılmıştır.

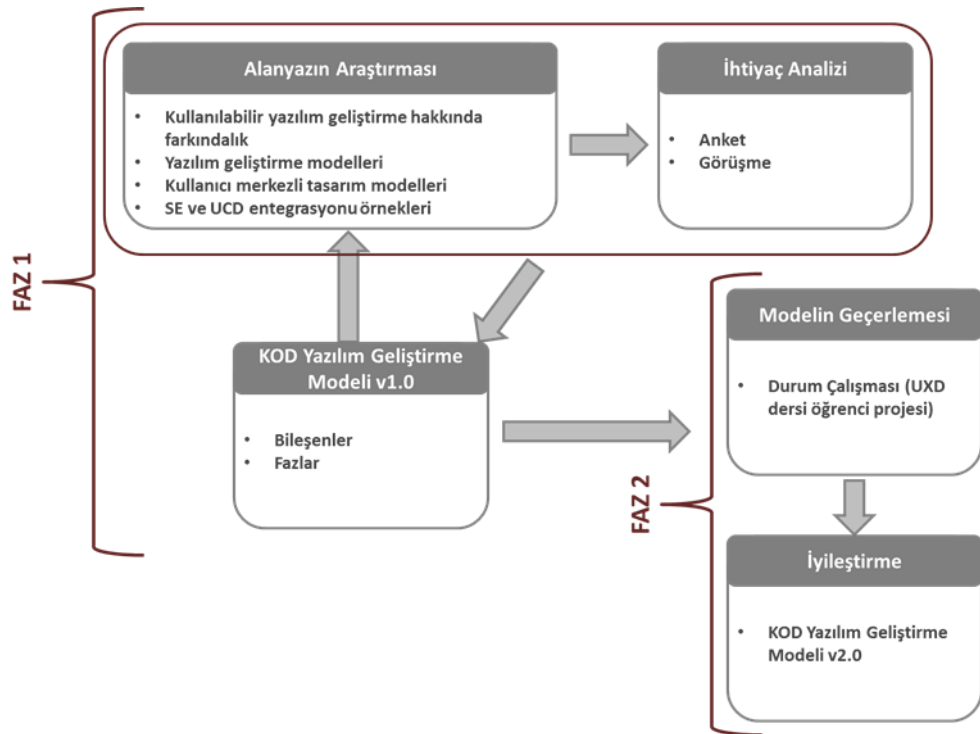


Şekil 2.1. Çeşitleme Tasarımı (Creswell ve diğ., 2007)

Çalışma kapsamında çeşitleme tasarım yönteminin amacına uygun olarak anket ile nicel veriler toplanmış, sonrasında görüşmeler ile nitel veriler toplanmış ve her ikisinden elde edilen bulgular beraberce değerlendirilerek çalışma kapsamında önerilen modelin temellendirilmesi sağlanmıştır.

2.2. Araştırma Tasarımı

Araştırmanın tasarımında tekrarlı ve artımlı bir yaklaşım izlenmiştir. Bu doğrultuda araştırma birbiriyle örtüşen ve döngüsel şekilde fazlar ve adımları arasında geribildirimler sağlanan iki ana fazda gerçekleştirilmiştir. Araştırmanın tasarımı Şekil 2.2'de görülmektedir.



Şekil 2.2. Araştırma Tasarımı

Araştırmanın birinci fazı kapsamında öncelikle çalışmanın amacı ile ilgili alanyazın araştırması gerçekleştirilmiştir. Bu araştırmalar sonucunda yazılım geliştirme modelleri, kullanıcı merkezli tasarım modelleri ve yazılım mühendisliği ve kullanıcı merkezli tasarım modellerinin entegrasyonuna yönelik olarak daha önce gerçekleştirilmiş çalışmalar incelenmiştir. Yine dünyada ve Türkiye’de kullanılabilirlik ve kullanıcı merkezli tasarıma yönelik faaliyetlerin yazılım geliştirme organizasyonlarındaki uygulanma durumlarına yönelik gerçekleştirilmiş çalışmalara ulaşılarak sonuçları değerlendirilmiştir.

Yine araştırmanın birinci fazında, alanyazın araştırmasına ek olarak, önerilen modelin ortaya çıkması için Türkiye’deki yazılım organizasyonlarının güncel yazılım geliştirme süreçleri ve kullanılabilirliğin bu süreçlere entegrasyonunun durumunu tespit etmeye yönelik bir anket çalışması yapılarak ihtiyaç analizi yapılmıştır. Nicel veri olarak değerlendirilen anket bulgularına ek olarak yine organizasyonlarda çeşitli pozisyonlarda çalışan kişiler ile birebir görüşmeler gerçekleştirilerek durumun detaylı analizleri gerçekleştirilmiştir. Anket ve görüşme bulguları modelin gelişimine katkı sağlamıştır.

Birinci fazın sonunda yazılım mühendisliği, İBE, çevik yaklaşımlar ile ilgili araç, teknik ve yöntemler ile ilgili yapılan alanyazın araştırmaları, anketler ve görüşmeler sonrasında mevcut durumda yaşanan kullanılabilirliğin sağlanması probleminin çözümüne yönelik olarak Kullanıcı Odaklı Davranışsal Yazılım Geliştirme Modeli (KOD Yazılım Geliştirme Modeli)’nin birinci sürümü geliştirilmiştir. KOD Sürüm1 (v.1)’de modelin fazları ve ana bileşenleri ortaya konmuştur.

Araştırmanın ikinci fazında, geliştirilen KOD v.1 Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Ana bilim dalında verilmekte olan User Experience Design (UXD - Kullanıcı Deneyimi Tasarım) (URL-3, 2014) adlı yüksek lisans dersinde öğrenciler tarafından dönem projeleri kapsamında uygulanmıştır. Böylece önerilen KOD v.1’in eksik taraflarının görülebilmesi ve uygulanabilirliğinin test edilmesi amaçlanmıştır. Dersin amaçları kapsamında modelin tüm aşamaları değil; yazılım geliştirme kısmı dışındaki aşamalarının uygulanması sağlanmış ve sonrasında öğrenciler ile görüşmeler yapılarak alınan geribildirimler doğrultusunda

modelde iyileştirmeler yapılmıştır. Böylece KOD Yazılım Geliştirme Modelinin son sürümü ortaya çıkarılmıştır.

2.3. Katılımcılar

Çalışmanın birinci fazında KOD v.1'in geliştirilmesi öncesinde Türkiye'deki kullanılabilirliğin yazılım geliştirme süreçlerine entegrasyonunun durum tespitine yönelik anket uygulanmış ve yazılım organizasyonlarında çeşitli pozisyonlarda görev alan geliştiriciler ve proje yöneticileriyle görüşmeler gerçekleştirilmiştir. KOD Yazılım Geliştirme son sürümünün geliştirme sürecinde ise önerilen model, Kullanıcı Deneyimi Tasarım dersi öğrencileri tarafından uygulanmış ve sonrasında modele ilişkin geribildirimlerle ilgili görüşme yapılmıştır.

Önerilen modelin ilk sürümünün oluşturulması öncesinde Türkiye'deki organizasyonlardaki durum tespitine yönelik olarak gerçekleştirilen anket çalışmasına toplamda 73 kişi katılmıştır. Katılımcıların mezun oldukları lisans bölümleri, eğitim durumları ve şirketlerde aldıkları pozisyonlar Tablo 2.1'de yer almaktadır. Katılımcıların görev aldıkları pozisyonlar incelendiğinde, büyük çoğunluğu yazılım ya da sistem geliştirici ve programcı olarak görev yaparken katılımcıların küçük bir kısmının kullanılabilirlik ile ilgili bir unvana (kullanılabilirlik uzmanı/mimarı/tasarımcısı/mühendisi) sahip olduğu görülmektedir.

Tablo 2.1. Anket katılımcılarının demografik bilgileri

	Kişi Sayısı
Eğitim Durumu	
Lisans	56
Yüksek Lisans	15
Doktora	2
Alan	
Bilgisayar Müh.	51
Elektronik H.Müh.	10
Matematik Müh.	2
Endüstri Müh.	2
Matematik	3
İşletme	3
Diğer (BÖTE, Finansal Matematik)	2
Unvanlar	
Yazılım sistem tasarımcısı/Geliştirici/programcı	44
İş analisti/Geliştiricisi	9
Sahip/Yönetici/Denetimsel pozisyon	5
BT Stratejisti/çalışanı/mühendisi	4
Kullanılabilirlik mühendisi/uzmanı/mimarı/tasarımcısı	2
Tasarımcı/ Arayüz tasarımcısı	1
Sistem yöneticisi/mühendis /mimar	1
Diğer	7

Yine birinci faz sırasında, anket çalışması sonrasında, yazılım geliştirme organizasyonlarında çalışan geliştiriciler ya da proje yöneticileriyle ile kullanılabilirliğin sağlanmasına yönelik olarak uyguladıkları yöntemlerle ilgili daha detaylı bilgi almak için yüz yüze görüşmeler yapılmıştır. Altı farklı organizasyondan toplam 8 kişi ile görüşme gerçekleştirilmiştir. Katılımcıların çoğunluğunu yazılım geliştiriciler ve test mühendisleri oluştururken yalnızca bir tanesi bulunduğu organizasyonda kullanılabilirlik uzmanı olarak görev almaktadır.

Katılımcıların çalıştıkları organizasyonların hizmet verdikleri alanlar ve proje geliştirdikleri sektör türlerine ait bilgiler de Tablo 2.2’de sunulmaktadır. Tablodaki katılımcı kodları oluşturulurken kurumları belirtmek için ‘K’ harfi, yazılım geliştirici için “YG” gibi şirketteki unvanların baş harfleri kullanılmıştır.

Tablo 2.2. Görüşme katılımcılarının demografik bilgileri

Katılımcı Kodu	Şirketteki unvan	Çalıştıkları şirketin hizmet verdiği alan	Proje geliştirdikleri sektör türleri
K1_YG_1	Yazılım Geliştirici (Araştırma Mühendisi)	Araştırma/ Bilim	E-imza kullanan tüm sektörlerle
K1_YG_2	Yazılım Geliştirici (Araştırma Mühendisi)	Araştırma/ Bilim	Kriptoloji/Güvenlik
K2_YG_3	Yazılım Geliştirici (Araştırma Mühendisi)	Teknoloji	Endüstriyel otomasyon
K3_UY_1	Yazılım Uzman Yardımcısı	Finans/Kamu Kurumu	Finans
K3_İA_1	İş Analisti	Finans/Kamu Kurumu	Finans
K4_KU_1	Kullanılabilirlik Uzmanı	Finans/Kamu Kurumu	Finans
K5_TM_1	Test Mühendisi	Danışmanlık	ERP projeleri
K6_TM_2	Test Mühendisi	Telekomünikasyon	Telekomünikasyon

Çalışmanın ikinci fazında geliştirilen modelin geçerlemesini yapmak, uygulanabilirliğini değerlendirmek amacıyla BLM 518 Kullanıcı Deneyimi Tasarımı dersi kapsamında yüksek lisans öğrencilerinin uyguladığı çalışmaya 3’er ve 2’şer kişilik iki takım halinde 5 kişi katılmıştır. Derse katılan öğrencilerin görev aldıkları pozisyonlar, kullanıcı deneyimi tasarımı konusunda önceki deneyimleri, çevik yazılım geliştirme yöntemleri konusundaki tecrübeleri arasında farklar bulunmaktadır. Katılımcıların hiç birinin test ya da davranış güdülü geliştirme konusunda deneyimi bulunmamaktadır. Bu farklar Tablo 2.3’te özetlenmektedir. Tablodaki katılımcı kodları oluşturulurken proje gruplarını belirtmek için ‘PG’ harfi, proje grubunda yer alan her bir öğrenciyi takım üyesi şeklinde belirtmek için “T” harfi kullanılmıştır.

Tablo 2.3. Model geerleme grşmesine katılanların demografik bilgileri

Kiřiler	Takımlar	Grev aldıkları pozisyonlar	Kullanıcı deneyimi tasarımı deneyimi	evik yazılım geleiřtirme tecrbesi	Test/Davranıř gdl geleiřtirme tecrbesi
PG1_T1	Proje Grubu 1	Uzman Arařtırmacı	Daha nce bu konuda bilgi edinmiř.	SCRUM konusunda bilgi sahibi	Yok.
PG1_T2		Arařtırma Mhendisi	Ders projeleri	Deneyim yok.	Yok.
PG1_T3		Bilgi İřlem	Deneyim yok	Deneyim yok.	Yok.
PG2_T1	Proje Grubu 2	Web Yazılım Geleiřtirici	Deneyim yok	SCRUM konusunda bilgi sahibi	Bilgisi var ama kullanmamıř.
PG2_T2		Yazılım Geleiřtirici	Deneyim yok	Yok.	Yok.

2.4. Değerlendirme Uygulaması

alıřmanın ikinci fazında geleiřtirilen modelin deėerlendirmesini yapmak, uygulanabilirliėini belirlemek amacıyla BLM 518 Kullanıcı Deneyimi Tasarımı dersi kapsamında bir geerleme alıřması geekleřtirilmiřtir. Kullanıcı Deneyimi Tasarım dersinde ėrencilere verilen proje kapsamı İBE prensiplerine dayalı olarak bir web arayz geleiřtirmektir. Bu projede hazırlanacak web arayz iin proje gruplarına aynı konu verilmiřtir. Verilen konu beslenme konusunda tketiciler arasında farkındalık yaratacak, tketicilerin eřitli zelliklerine (alerjik durum, hastalık vb.) gre onlara satın alacakları besinler hakkında bilgi ve uyarılar sunacak olan bir web uygulamasının arayzlerini geekleřtirmektir. Proje kapsamında ėrencilerin potansiyel geek kullanıcıları gzlemleyerek gereksinimleri belirlemeleri ve sonrasında onlar ile geleiřtirdikleri arayz prototiplerini kullanılabilirlik deėerlendirme yntemlerini kullanarak deėerlendirmeleri ve sonularını sunmaları beklenmektedir.

Proje kapsamında geleiřtirilen uygulamanın geeklenmesi beklenmemektedir. Bu nedenle KOD Yazılım Geleiřtirme Modelinin, kodlama iermeyen adımları bu proje kapsamında uygulanabilmiřtir. Derse katılım 5 kiři ile saėlandıėından 2 takım oluřturulmuřtur. İki proje grubu; Proje Grubu 1 ve Proje Grubu 2 alıřmalarını aynı řartlarla, birbirlerinden ayrı srdrmřlerdir. ėrencilere dnemin bařında proje alıřmalarına bařlamadan nce KOD Yazılım Geleiřtirme Modeli aıklanmıřtır ve ėrencilere dnem boyunca dersin web sayfasından evrimii destek saėlanmıřtır.

Yüksek Lisans dersi bir dönemlik olduğundan proje süresi bu ders kapsamında yaklaşık 1,5 ay kabul edilmiştir. Projenin fazları zaman-tanımlı olarak toplam proje süresi gözetilerek 2 haftalık yinelemeler olacak şekilde planlanmıştır. 2 haftada bir izlenen öğrencilerin proje süresince uygulaması gereken adımlar Tablo 2.4’te gösterilmektedir.

Tablo 2.4. Uygulanacak adımlar ve zaman çizelgesi

Zaman	Proje fazı	KOD Değerlendirme Amacı
1.Hafta	Gerçek kullanıcılardan gereksinim toplama	Gereksinim toplama kolaylığı, gerçek kullanıcılara kolay erişebilme
3.Hafta	Alınan gereksinimlerin KOD senaryolarına dönüştürülmesi	DOG(BDD) yaklaşımının kullanım zorluğu/kolaylığı
5.Hafta	Yazılım araçlarıyla gereksinimlerin yansıtıldığı ekran tasarımlarının yapılması	Erken aşamalarda sistemin prototip oluşturabilme yeteneği, KOD senaryolarının kullanılarak tasarımların oluşturulabilme
7.Hafta	Tasarlanan ekranların KOD senaryoları yöntemiyle kullanılabilirlik testlerinin yapılması	zorluk/kolaylık seviyesi Tasarlanan ekranların kullanılabilirlik açısından başarısı

2.5. Veri Toplama Araçları

KOD Yazılım Geliştirme Modelinin geliştirilmesi sürecinde birinci faz kapsamında veri toplama aracı olarak anket ve görüşmelerden faydalanılmıştır. Yine ikinci faz kapsamında modelin değerlendirilmesi sürecinde, proje takımlarından geri bildirim almak amacıyla da yine görüşme yönteminden faydalanılmıştır.

Birinci fazın başında gerçekleştirilen anket çalışması kapsamında Türkiye’deki yazılım geliştirme organizasyonlarında uygulanan kullanılabilirlik çalışmalarının durumunu belirlemeye yönelik olarak 32 soruluk çoktan seçmeli ve likert ölçeğinde sorular hazırlanmıştır. Anket soruları aşağıdaki dört başlık altında gruplandırılmaktadır.

- Katılımcılar ve organizasyonları
- Kullanılabilirlik kavramının anlaşılması ve uygulanması

- Kullanılabilirlik deęerlendirmelerinin gerekleřtirilmesi
- Yazılım geliřtirme profesyonellerinin kullanılabilirlik algıları

Anket katılımcıları belirlenirken farklı sektörlerde alıřanların ve farklı bölümlerde eğitim gören öęrencilerin katılımı dikkate alınmıřtır. Ayrıca anket web üzerinden yapıldığı için katılımcılara coęrafi konum kısıtı getirilmemiřtir. Böylece katılımcıların geniř spektrumdan seilmesi için özen gösterilmiřtir.

Anket soruları hazırlanırken alanyazındaki benzer alıřmalardan faydalanılmıř, ilgili sorular Türke’ye evrilerek adapte edilmiřtir. Hazırlanan sorulardan 16, 17, 18, 26 ve 27 numaralı sorular Humayoun, Dubinsky ve Catarci (1999)’dan; 28, 29 ve 31 numaralı sorular ise Gulliksen (2004)’in alıřmasından Türkeye evrilmiřtir. Hazırlanan sorular, konu ile ilgili daha önce anket tasarlamıř ve bu konuda eğitim veren kiřilere gözden geirmeleri için iletilmiř ve verdikleri geribildirimler doęrultusunda güncellemeler yapılmıřtır. Anket soruları Ek-A’da yer almaktadır.

Yine birinci faz kapsamında Türkiye’deki kullanılabilirlik alıřmalarının durumunu daha detaylı inceleyebilmek amacıyla farklı řirketlerde farklı pozisyonlarda alıřan kiřiler ile yarı-yapılandırılmıř görüřmeler gerekleřtirilmiřtir. Yapılan görüřmelerde de ankette belirlenen 4 bařlık üzerinden sorular yöneltilmiřtir. Görüřme sorularına Ek-B’den ulařılabilmektedir.

Görüřme katılımcıları belirlenirken, coęrafi açıdan kolay ulařılabilmesi açısından Kocaeli ve İstanbul’da hizmet veren řirketler arařtırılmıřtır. Kullanıcı deneyimi ve yazılım yařam süreçleriyle ilgili bilgi alınabileceęi düşünölen bölümlerde alıřan kiřilerle görüřölmüřtür. Toplanacak bilgilerin eřitlilięi arttırması adına kiřilerin ayrı bölümlerde alıřıyor olmasına dikkat edilmiřtir. Geliřtirme sürecinin ikinci fazında, önerilen modelin geerlemesinin saęlanması için Kullanıcı Deneyimi Tasarım dersini alan yüksek lisans öęrencilerinin projelerinde modeli uygulamaları saęlanmıřtır. Modelin ilk sürümü için yapılan görüřmelerde sorular mevcut modellerdeki olası problemlerin ve bu problemlere özüm önerilerinin tespitine yönelik hazırlanmıřken, ikinci sürüm için yapılan görüřmelerde sorular KOD Yazılım Geliřtirme Modeli’ne iliřkin geribildirimleri almak amacıyla hazırlanmıřtır.

Geçerleme çalışması için Kullanıcı Deneyimi Tasarımı dersi alan öğrencilerin seçilmesinin sebeplerinden en önemlisi kullanıcı deneyimi konusunda teorik bilgi sahibi olmaları ve bu konuda ilgilerinin olmasıdır. Bu yüzden onlardan gelecek geribildirimlerin önemli olacağı ve modele katkı sağlayacağı düşünülmüştür.

Görüşme soruları 5 ana başlık altında toplanmıştır. Bu görüşmeye ait sorular Ek-C’de yer almaktadır.

- Katılımcılar ve çalıştıkları organizasyonlarda görev aldıkları pozisyonları
- Kullanıcı deneyimi tasarımı konusunda tecrübeleri
- Çevik yazılım geliştirme konusunda tecrübeleri
- KOD yazılım geliştirme fazları
- DOG(BDD) ve Kullanıcı Odaklı Tasarım

2.6. Veri Analizi

İhtiyaç analizine yönelik yapılan anket, tanımlayıcı istatistik yöntemi ile incelenmiştir. Anket cevapları standart bir formatta olduğu için sonuçlar analiz edilirken soru tipine göre sıklık ve yüzdelik oranlar kullanılarak değerlendirilmiştir.

Görüşmeler sırasında katılımcılara büyük ölçüde açık uçlu, yoruma dayalı sorular yöneltilmiştir. Görüşme cevaplarının analizi sırasında klasik içerik analizi yöntemi uygulanmıştır. Klasik içerik analizi yönteminde temalar ve kategoriler sadece veriye bakılarak oluşturulmaktadır (Hsieh ve Shannon, 2005). Temalar görüşme soruları ile de örtüşecek şekilde kullanılabilirlik kavramının anlaşılması ve uygulanması, kullanılabilirlik değerlendirmelerinin gerçekleştirilmesi, yazılım geliştirme profesyonellerinin kullanılabilirlik algıları şeklinde üç ana grupta tanımlanmıştır.

Katılımcıların verdikleri cevaplara dayalı olarak da bu temalar atında Tablo 2.5’te yer alan alt kategoriler ve temalar belirlenmiştir. Her kategori kodlanmıştır. İçerik analizinin katılımcıların verdikleri yanıtlara dayalı olan tüm detayları Ek-D’de paylaşılmaktadır.

Tablo 2.5. Görüşme analizi kategorileri

Tema	Kategori kodu	Kategori tanımlaması
Kullanılabilirlik kavramının anlaşılması ve uygulanması	KG_1_1	İBE ya da kullanılabilirlik konularıyla ilgili personel
	KG_1_2	Kullanılabilirliğin sağlanmasına verilen önem
	KG_1_3	Organizasyonlarda uygulanan yazılım yaşam döngüsü
	KG_1_4	Kullanılabilirliğin yaşam döngüsüne entegrasyonu
	KG_1_5	Uygulanan modelleme yaklaşımları
Kullanılabilirlik değerlendirmelerinin gerçekleştirilmesi	KG_2_1	Kullanılabilirlik gereksinimlerinin toplanması
	KG_2_2	Kullanılabilirlik yöntemlerini uygulama
	KG_2_3	Kullanılabilirlik değerlendirme sonuçları
Yazılım geliştirme profesyonellerinin kullanılabilirlik algıları	KG_3_1	Kullanılabilirlik değerlendirmesinin yazılım geliştirme sürecinde uygulanmasının avantajları ve dezavantajları
	KG_3_2	Kullanılan yazılım yaşam döngüsüne kullanılabilirlik entegrasyonu
	KG_3_3	Yazılım yaşam döngüsüne kullanılabilirlik entegrasyonunda uygulanabilecek aktiviteler

Modelin geçerlenmesi sonrasında öğrenciler ile yapılan görüşme cevaplarının analizi sırasında da klasik içerik analizi yöntemi uygulanmıştır. Temalar görüşme soruları ile de örtüşecek şekilde kullanıcı deneyimi konusunda farkındalık, çevik yöntemler konusunda tecrübeler, KOD Analiz ve tasarım safhası, DOG(BDD) yaklaşımı ve kullanıcı odaklı tasarım şeklinde dört ana grupta tanımlanmıştır. Katılımcıların verdikleri cevaplara dayalı olarak da bu temalar altında Tablo 2.6’da yer alan alt-kategoriler belirlenmiştir. İçerik analizinin katılımcıların verdikleri yanıtlara dayalı olan tüm detayları Ek-E’de paylaşılmaktadır.

Tablo 2.6. Kullanıcı deneyimi tasarım dersi projesi öğrenci yorumları

Tema	Kategori kodu	Kategori tanımlaması
Kullanıcı deneyimi konusunda farkındalık	KG_1	Kullanıcı deneyimi konusunda tecrübeler
Çevik yöntemler konusunda tecrübeler	KG_2_1	Kullanılan çevik yöntemler
	KG_2_2	Çevik yöntemlerden TOG/DOG (TDD/BDD) hakkında tecrübe
KOD Analiz & Tasarım safhası	KG_3_1	Ders projesinde gerçekleştirilen süreç ve takım çalışması
	KG_3_1	DOG (BDD)’nin sürece katkısı
DOG(BDD) yaklaşımı ve kullanıcı odaklı Tasarım	KG_4	DOG (BDD)’nin kullanıcı odaklı tasarıma katkısı

2.7. Çalışmanın Güvenilirliği

Yapılan çalışma kapsamında anket ile nicel veriler, görüşmeler ile nitel veriler toplanmış ve ikisinden elde edilen bulgular beraber değerlendirilerek çalışma kapsamında önerilen modelin temellendirilmesi sağlanmıştır. Nicel araştırma için geçerli olan güvenilirlikle ilgili bazı etkenler, nitel araştırma için söz konusu değildir. Örneğin, nitel araştırmanın temel özelliklerinden biri olan algıların önemi ve doğal ortama duyarlılık, güvenilirlik konusunda bazı sorunlar ortaya çıkarmaktadır (Yıldırım, 2010). İnsan davranışları her zaman durağan olmadığından, gerçekler bireylere ve içinde bulunulan ortama göre sürekli bir değişim içinde olduğundan dolayı yapılan nitel araştırmanın benzer gruplarda tekrarlanması aynı sonuçları vermeyecektir.

Güvenilirlik çalışmaları nicel araştırmalarda kolayca uygulanabilir. Tekrar tekrar uygulandığında da yine aynı sonuç ve cevapları üretecektir. Ancak nitel çalışmalarda aynı durum söz konusu değildir. Nitel çalışmalarda niteliği arttırmak için literatürde yer alan kullanılabilecek stratejiler aşağıdaki gibidir (Yıldırım, 2010):

- Katılımcı Teyidi (Dönüt alma, İletişimsel Geçerleme)
- Uzun Süreli Çalışmalar
- Ayrıntılı Betimleme
- Kısa Yoldan Denetleme
- Eş Denetleme (Uzman İncelemesi)
- Çeşitleme (Üçgenleme, Sac Ayağı)
- Negatif ve Alternatif Sonuçlarla Karşılaştırma
- Çarpık Durumları Gösterme
- Kendini Değerlendirme (Dönüşlülük, Eylemden Etkilenme)
- Araştırmanın Sınırlarını Ortaya Koyma
- Analiz Formu
- Bağlantılara Yoğunlaşma
- Ayrıntılı Alıntılar Yapma
- Rastlantısal Örneklem Seçme
- Tekrarlı Sorular
- Sorularla Araştırmacının Kendi Rolünü Sorgulaması

Bu alıřma kapsamında eřitleme yntemi kullanılarak farklı veri toplama aralarından anket ve grüşmelerin bir arada kullanımı saėlanmıřtır. Veri toplama araları hazırlanırken uzman incelemesi yntemi uygulanmıřtır. Anket ve grüşme soruları hazırlanırken alanında uzman olan kiřilerin (hem anket hazırlama konusunda hem de YM ve İBE alanında uzman farklı kiřilerce) kontrol etmesi saėlanmıřtır. Saėladıkları geribildirimler doėrultusunda sorularda dzenlemeler yapılmıřtır. Elde edilen verilerin doėruluėunun onayı iin de katılımcı teyidi yntemi uygulanmıřtır. Yapılan grüşmeler sonrasında katılımcıların cevapları kâėıda dklp, katılımcılardan teyit almak iin onlara geri gnderilip onayları alınmıřtır.



3. ANALİZ VE DEĞERLENDİRME BULGULARI

Bu bölümde araştırma kapsamında birinci faz sırasında gerçekleştirilen anket çalışması ve yapılan görüşmeler, önerilen modelin ilk sürümü ve son sürümünün geliştirimi sırasında yapılan görüşmelerden elde edilen bulgular paylaşılmaktadır. Anket çalışmasına ait bulgular tanımlayıcı istatistik ile analiz edilmiş olup bulgular sıklık ve yüzdelik oranlar ile sunulmaktadır. Görüşme bulguları ise içerik analizine tabi tutulmuş olup, oluşturulan kategoriler altında katılımcıların ifadeleri ile desteklenerek sunulmaktadır.

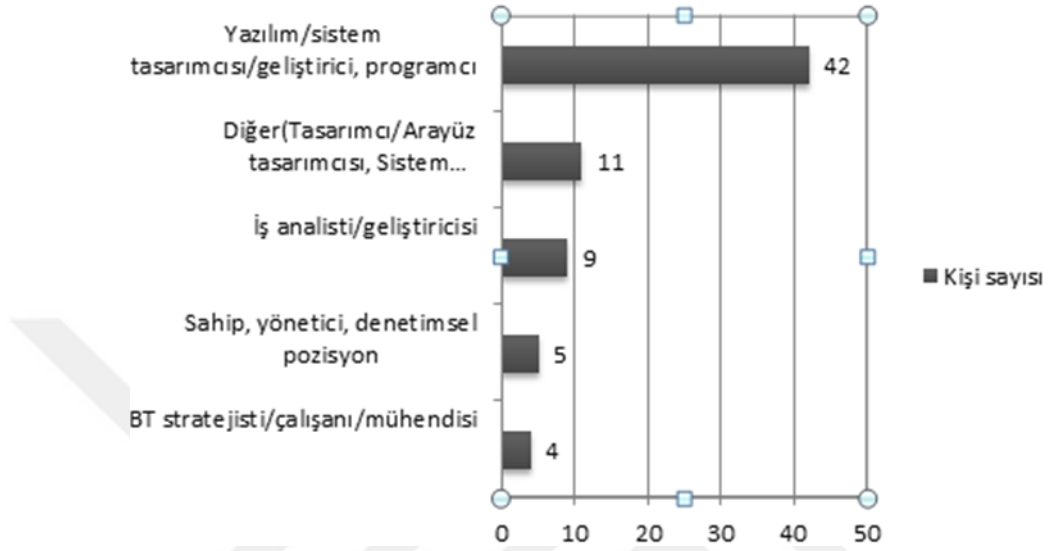
3.1. Birinci Faz: Anket Çalışmasının Bulguları

KOD Yazılım Geliştirme modelinin ilk sürümünün geliştirimi öncesinde, mevcut modellerin eksiklerini görmek ve analizini yapabilmek adına Türkiye’de faaliyet gösteren yazılım organizasyonlarında uygulanan yazılım geliştirme süreçleri ve kullanılabilirlik değerlendirmelerinin bu süreçlerde ne kadar uygulandığını sorgulayan bir anket çalışması yapılmıştır. Bu bölümde bu anket çalışmasının bulguları katılımcı özellikleri ve organizasyon bilgileri, kullanılabilirlik kavramının anlaşılması ve uygulanması, kullanılabilirlik değerlendirmelerinin gerçekleştirilmesi ve yazılım geliştirme profesyonellerinin kullanılabilirlik algıları şeklinde tanımlanan başlıklar altında açıklanmaktadır.

3.1.1. Katılımcı ve organizasyon bilgileri

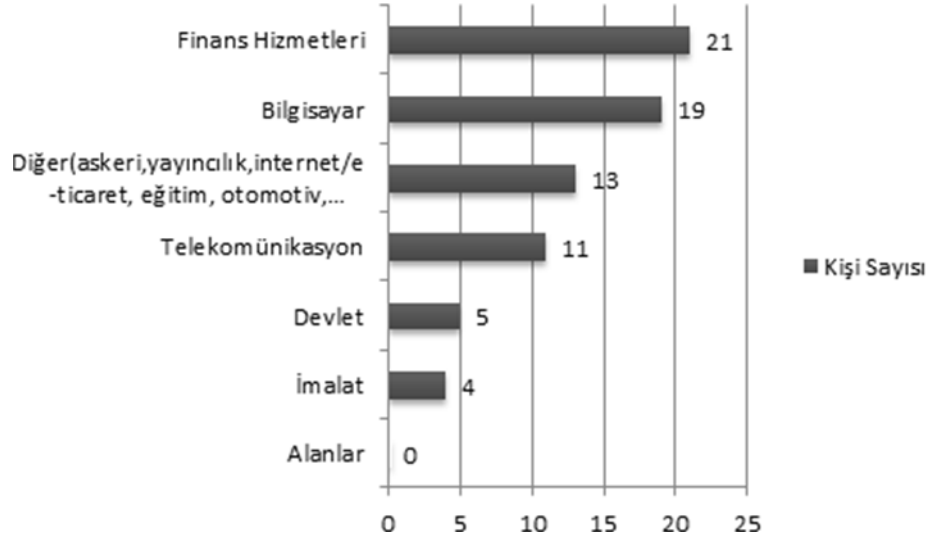
Anket çalışmasına katkıda bulunan katılımcıların öncelikle çalıştıkları organizasyonlarda görev aldıkları pozisyonlar incelenmiştir. Daha sonra katılımcıların çalıştıkları organizasyonları ile ilgili detaylı bilgiler sorgulanmıştır. Organizasyonların hangi sektörlerden olduğu, faaliyet süreleri, organizasyon içerisinde uyguladıkları yazılım yaşam döngüsü olup olmadığı ve bunun ne olduğuna yönelik sorular yöneltilmiştir. Katılımcıların, çoğu çalıştıkları organizasyonlarında yazılım ya da sistem geliştiricisi ve programcı olarak görev yapmaktadır.

Katılımcıların çok küçük bir bölümünün kullanılabilirlik ile ilgili bir unvana (kullanılabilirlik uzmanı/ mimarı/ tasarımcısı/ mühendisi) sahip olduğu görülmektedir. Katılımcıların sahip oldukları unvanların dağılımı Şekil3.1’de yer almaktadır.



Şekil 3.1. Anket katılımcılarının unvan dağılımları

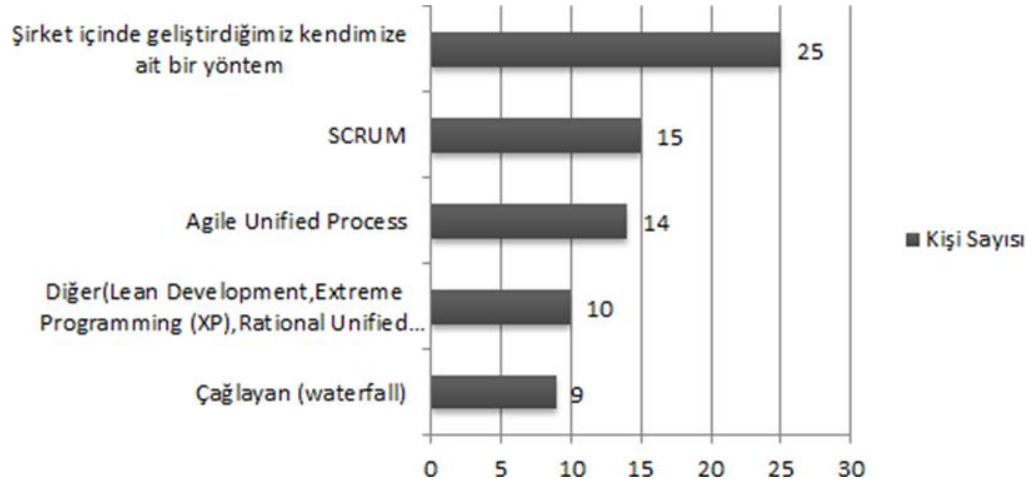
Katılımcıların görev aldıkları organizasyonların yer aldığı sektörlerin önemli bölümünü Şekil 3.2’de görüldüğü üzere, sırasıyla finans-bankacılık, bilişim ve telekomünikasyon oluşturmaktadır. 4 katılımcıdan oluşan diğer başlığı altında yayıncılık, internet/e-ticaret, eğitim, otomotiv ve havacılık sektörleri yer almaktadır. Bu açıdan anket katılımcılarının sektörel çeşitliliği, önerilen modelin farklı alanlarda kullanılabilir olmasına katkı sağlamıştır.



Şekil 3.2. Anket katılımcılarının sektörel dağılımları

Çalışmadaki katılımcıların yer aldığı organizasyonların büyük bir çoğunluğu 50 kişinin üzerindeki büyük ölçekli firmalardan oluşmaktadır. Organizasyonların sektörde var olma sürelerine bakıldığında ise yarısından azı, 1-10 yıl deneyime sahipken geri kalanı 10 yıl ve üzeri deneyime sahiptir. Bu sonuçlar çalışmaya katılan organizasyonların çoğunlukla finans/bankacılık alanından gelmesine bağlıdır.

Katılımcılara, organizasyonlarında uyguladıkları bir yazılım yaşam döngüsü modeli olup olmadığı sorulmuştur. Katılımcılardan çoğu bir model uyguladıklarını bir kısmı ise herhangi bir standart yaklaşımları olmadığını, her projede farklı yaklaşımları takip ettiklerini belirtmişlerdir. Bir yaşam döngüsü modeli kullananlar arasında da en yaygın kullanılan yöntemlerden şirket içinde geliştirilen kendilerine ait bir yöntem olduğunu söylemişlerdir. Şekil 3.3'te katılımcıların uyguladıkları yazılım yaşam döngüsü modeli dağılımı gösterilmektedir.



Şekil 3.3. Kullanılan yazılım yaşam döngüsü modeli

3.1.2. Kullanılabilirlik kavramının anlaşılması ve uygulanması

Yapılan ankette kullanılabilirlik kavramının ne kadar anlaşıldığı ve uygulandığını belirlemeye yönelik katılımcılara sorular sorulmuştur. Katılımcılardan çoğu İBE personeline sahip olduklarını belirtmişlerdir. Bu personellerin İBE hakkında tecrübeyi nasıl kazandıkları sorulduğunda ise katılımcıların çoğu ilgili personelin konu hakkında resmi bir program ya da ders kapsamında tecrübe sahibi olduğunu düşünmektedir. Ancak katılımcılardan azımsanmayacak bir kısmının kullanılabilirlikten sorumlu personelin kurum içerisindeki unvanının “arayüz tasarımcısı” olarak tanımlaması bu konuda yanlış bir algının olabileceğinin göstergesidir. Organizasyonlarda İBE ve kullanılabilirlik kavramlarının anlaşılma ve uygulanma durumunu gösteren bilgiler Tablo 3.1’de yer almaktadır.

Tablo 3.1. İBE/Kullanılabilirlik kavramının organizasyonlarda anlaşılması ve uygulanması

	Kişi Sayısı
İBE ile ilgili eğitilmiş personele sahip olma durumu	
Evet	23
Hayır	6
Bilmiyorum	11
İBE ile ilgili eğitime sahip personelin konu hakkındaki bilgiyi edinme kaynakları	
İBE veya kullanılabilirlik konusunda bir derece programı ile	14
İBE ile ilgili bir ders ile	13
Kendi kendilerine	10
İş başında deneyimle	12
Bilmiyorum	15

Tablo 3.1. (Devam) İBE/Kullanılabilirlik kavramının organizasyonlarda anlaşılması ve uygulanması

Kullanılabilirlikten sorumlu personel unvanı (çoklu seçim)	
Kullanıcı Deneyimi Tasarımcısı	14
Bilgi Mimarı	9
Kullanılabilirlik Mühendisi	6
Arayüz Tasarımcısı	22
Ürün Tasarımcısı	11
Kullanıcı Deneyimi Mimarı	6
Kullanılabilirlik Uzmanı	8
Diğer(Kullanılabilirlik Profesyoneli, Kullanıcı Araştırmacısı, vb.)	25

Kullanılabilirlik ile ilgili yöntemlerin yazılım yaşam döngüsü boyunca hangi evrelerde kullanıldığına dair sorulara katılımcıların verdikleri cevaplar Tablo 3.2’de gösterilmektedir. Tablo 3.2’den de anlaşıldığı üzere projelerde kullanılabilirlik yöntem ve tekniklerinden kullanılabilirlik testlerinin, sezgisel değerlendirme, kontrol listelerinin, memnuniyet anketlerinin en çok test basamağında uygulandığı görülmektedir. Bu da kullanılabilirliğin değerlendirilmesine yönelik yöntemlerin son aşamalarda daha yoğun kullanıldığını göstermektedir.

Tablo 3.2. Kullanılabilirlik yöntem ve tekniklerinin uygulandığı yazılım geliştirme aşamaları

	Ön-çalışma	Gereksinim	Tasarım	Yapım (Kodlama)	Test	Bakım
Kullanım Durumu (Use Case)	12	12	7	0	2	1
Hikâye panosu (Storyboard)	1	13	12	0	0	1
Vizyon/kapsam dokümanı	11	10	4	2	1	1
Grafik Arayüz Belirtilimleri	1	4	21	4	1	1
Kâğıt ya da yazılım prototipler	0	5	16	9	0	1
Müşteri görüşmeleri	16	9	3	2	1	1
Kullanılabilirlik testleri	0	3	4	3	21	1
Sezgisel (heuristic) değerlendirme	2	2	6	5	11	1
Kontrol listeleri	1	3	4	3	18	1
Stil Kılavuzları	1	5	15	1	5	2
Görev analizi	9	7	11	0	2	1
Memnuniyet anketleri	0	1	3	0	7	20
Persona/kullanıcı profilleri	3	7	6	4	4	3
Kıyaslama (benchmarking) çalışmaları	6	3	5	1	6	8
Görsel tasarım	2	2	19	5	1	2
Kullanım loglarının analizi	1	2	4	1	6	17

Kullanılabilirliğin uygulanmasının önündeki engellerin neler olabileceği konusunda katılımcılara soru yöneltmiş ve bu soruda katılımcılara çoklu seçim hakkı

tanınmıştır. Katılımcılar kullanılabilirliğin uygulanması önündeki engellerin en başında zamanın yeterli gelmemesi ve proje ekibinin sayıca yetersiz olmasını belirtmişlerdir. Diğer nedenler ve katılımcıların bunlara ait görüşlerine ait bulgular Tablo 3.3'te görülmektedir.

Tablo 3.3. Kullanılabilirliğin uygulanamamasının sebepleri (çoklu seçim)

	Kişi sayısı
Zamanın yeterli olmaması	19
Proje ekibinin sayıca yetersiz olması	16
Bütçe ayrılmaması	10
Konu ile ilgili uzman personelin bulunmaması	9
Ekonomik ihtiyacın olmaması müşterinin istememesi	9
Kullanılabilirliğin şirket içerisinde yeterince bilinmemesi	6
Ne faydasının olacağının hesaplanamaması	6
Yönetim ya da mühendislerin kullanılabilirliği gerekli görmemesi	3

Katılımcılara geliştirdikleri projelerde kullanılabilirliğin sağlanmasının önemli bir konu olup olmadığı sorulduğunda katılımcıların çoğu evet cevabını vermiştir. Kullanılabilirliğin proje başarısında çok önemli olduğunu belirten katılımcıların çalıştıkları organizasyonların faaliyet alanına bakıldığında, yarısının bilişim sektöründe yer aldıkları görülmektedir. Aynı şekilde bu konunun önemli olduğunu vurgulayan katılımcılar çoğu daha çok 200'ün üzerinde çalışana sahip banka ve finans kuruluşları gibi büyük ölçekli organizasyonlarda görev almaktadırlar.

3.1.3. Kullanılabilirlik değerlendirmelerinin gerçekleştirilmesi

Organizasyonların kullanılabilirlik değerlendirmeleri sırasında katılımcıları nasıl ve kimlerden seçtiklerine, testleri ne kadar kişi ile gerçekleştirdiklerine ve hangi yöntemleri daha çok kullandıklarına yönelik sorular yöneltilmiştir. Uygulanan kullanılabilirlik testlerinde, katılımcıların çoğu testlerini temsili bir örneklem ile gerçekleştirdiklerini belirtirken, bir kısmı ise bu testleri çalışanları ile gerçekleştirdiğini belirtmiştir. Kullanılabilirlik testlerinde yer alan katılımcısı sayısının büyük bir oranla 0-5 kişi ile sınırlı kaldığı sonucuna ulaşılmıştır.

Kullanılabilirlik değerlendirmesinde hangi yöntemlerin organizasyonlarda sıklıkla kullanıldığı sorgulanmıştır. Tablo 3.4'te özetlenen sonuçlara bakıldığında tekniği bilmediğini söyleyenler ile o tekniği kullanmayanların sayılarının diğerlerine göre daha çok olduğu görülmektedir. Sıklıkla kullanılan yöntemlerin kullanıcı çalışmaları, kullanıcı testleri ve alan araştırmalarının olduğu ancak diğer yöntemlerin fazla bilinmediği ortaya çıkmaktadır.

Tablo 3.4. Kullanılabilirlik değerlendirme yöntemlerini kullanma sıklığı

		Hiçbir zaman	Nadiren	Sıkça	Genellikle	Her zaman	Bilmiyorum
Kullanılabilirlikle ilgili plug-in kullanmak	7		4	2	11	0	12
ISO 13407 gibi genel kullanılabilirlik çerçevelerini kullanmak	7		3	7	4	1	14
Genel İBE yöntemlerini kullanmak (Kullanıcı çalışmaları, kullanıcı testleri, alan araştırmaları, vb.)	1		4	4	12	10	7
Sezgisel değerlendirme gibi uzman tabanlı yöntemleri kullanmak (Nielsen'in kullanılabilirlik sezgiselleri gibi)	8		5	4	3	1	16
Stil Kılavuzlarına dayalı değerlendirme yapmak (Apple Tasarım Prensipleri gibi)	10		5	0	4	4	14

3.1.4. Yazılım geliştirme profesyonellerinin kullanılabilirlik algıları

Yazılım geliştirme profesyonellerinin kullanılabilirlik ile ilgili algıları Tablo 3.5'te sunulmaktadır. Katılımcılar, kullanılabilirliğin proje başarısı için önemli olduğuna inanmalarına rağmen maliyet artırıcı olduğunu da düşünmektedirler. Kullanılabilirliğin yazılım geliştirme yöntemine entegre olması ve organizasyonlarda ilgili personelin bulunması gerektiğine inanırken özellikle çevik yöntemler ile daha iyi entegre olacağını belirtmektedirler.

Tablo 3.5. Yazılım geliştirme profesyonellerinin kullanılabilirlik ile ilgili algıları

	Katılmıyor	Kararsız	Katılıyor
Kullanılabilirlikle ilgili gereksinimler projelerin başarısı için çok önemlidir	3	2	28
Kullanılabilirlik başlangıçtan itibaren proje planının bir parçasıdır	3	5	25
Kullanılabilirlik proje geliştirimi sırasında zaman kaybıdır.	26	4	3
Kullanılabilirlik şirketlere ekstra maliyet yüklemektedir.	8	9	16
Proje geliştirimi sırasında kullanılabilirlik açısından kullanıcıların sürece dahil olması önemlidir.	2	3	27
Kullanılabilirlik testleri projenin başarısı için çok önemlidir	2	6	24
Kullanılabilirliğin sağlanmasına yönelik aktiviteler şirket içerisinde kullandığımız Yazılım Geliştirme Yöntemine iyi entegre olmuştur	14	13	5
Şirketimizde kullanılabilirlik konusu ile ilgili farkındalık yeterince vardır	14	14	5
Şirketimizde kullanılabilirlikle ilgili aktiviteler üst yönetim tarafından da desteklenmektedir	12	17	2
Kullanılabilirliğin sağlanması projelerin maliyetini arttırıcı bir unsurdur	7	10	16
Kullanılabilirlikle ilgili aktiviteleri gerçekleştirmek için şirket içerisinde konu ile ilgili resmi bir eğitim almış kişilerin bulunması gereklidir	7	9	17
Kullanılabilirlikle ilgili aktivitelerin çevik yöntemlere daha uygun olduğunu düşünüyorum	6	11	16

3.1.5. Anket Değerlendirmesi

Bu çalışma sonuçları ortaya koymaktadır ki, yazılım geliştiren organizasyonlarda katılımcılar kullanılabilirliği uyguladıklarını düşünmektedirler. Ancak kullanılabilirlikle ilgili yöntemler hakkında bilgi sahibi olma oranı oldukça düşüktür. Kullanılabilirlikle ilgili farkındalığın daha çok büyük ölçekli organizasyonlarda olduğu görülmektedir. Katılımcılar kullanılabilirlik konusunu yazılım projelerinin başarısı için önemli bir gereksinim olarak değerlendirirler de organizasyonlar için ek maliyet ve zaman gerektireceğine inanmaktadırlar. Kullanılabilirlik yöntemlerinin yazılım yaşam döngüsüne çoğunlukla geç aşamalarda dahil edildiği görülmektedir ve çevik yaklaşımlar ile entegrasyonunun daha kolay olacağı görüşü hakimdir.

Alanyazın arařtırmaları ve yapılan anket alıřması sonrasında evik yaklařımların kullanılabilirlięe entegrasyonuna daha elveriřli oldukları sonucuna varılması nedeniyle, onerilecek modelin evik yaklařımlara daha yakın olması gerektięi sonucuna varılmıřtır. Ayrıca sistemin kullanılabilirlięinin sadece yařam dsnn sonunda test edilen bir kavram olarak dřnlmesinin yanlıř olduęunun ve onerilecek modelde bundan kaınılması gerektięi dřnlmřtr.

3.2. Birinci Faz: Grřme Bulguları

KOD Yazılım Geliřtirme Modelinin problem analizi iin anket uygulamasına ek olarak durumun detaylı analizinin yapılabilmesi iin eřitli alanlarda hizmet veren organizasyonlarda farklı pozisyonlarda grev alan, alanlarında uzman kiřilerle yazılım geliřtirme yntemleri ve kullanılabilirlięin entegrasyonuna ynelik yz yze grřmeler yapılmıřtır. Bu blmde, gerekleřtirilen grřmelere ait bulgular belirlenen temalar ve onlara ait alt kategoriler altında paylařılmaktadır.

3.2.1. Kullanılabilirlik kavramının anlařılması ve uygulanması

Katılımcılara alıřtıkları organizasyonlarda kullanılabilirlik kavramının ne kadar anlařıldığını ve uygulandığını ğrenmek adına sorular yneltirmiřtir. Alınan yanıtlardan elde edilen bulgular daha ok organizasyonlardaki İBE ya da kullanılabilirlik konularıyla ilgili personel (KG_1_1), kullanılabilirlięin saęlanması verilen nem (KG_1_2), organizasyonlarda uygulanan yazılım yařam ds (KG_1_3), kullanılabilirlięin yařam dsne entegrasyonu (KG_1_4) ve uygulanan modelleme yaklařımları (KG_1_5) konularındadır.

3.2.1.1. İBE ya da kullanılabilirlik konularıyla ilgili personel

Organizasyonlarda İBE ya da kullanılabilirlik konularıyla ilgili grevli personel bulunması konusunda katılımcıların byk oęunluęu olumsuz yanıtlar vermiřlerdir. Katılımcılardan K1_YG_2 bu konuda “Uzman/Uzman Yardımcısı kadrosunda, ara yz geliřtirmesi yapmıř yazılımcı” řeklinde tanımlayarak grevli bir personel olduęunu belirtmiřtir. Ancak bu katılımcı ile aynı organizasyonda grev yapan bir dięer katılımcı (K1_YG_1) ise bu grevde kimsenin yer almadığını test mhendislerinin bu sorumluluęu yerine getirdiğini belirtmiřtir. Bu farklılıęın nedeni, K1_YG_2 ile K1_YG_1 aynı organizasyonun farklı blmlerinde grev almasından

kaynaklıdır. Aynı organizasyon içinde bile unvanlar konusunda farklı anlayışlar görülebilmektedir. K4_KU_1’de organizasyonunda kullanılabilirliğin sağlanmasıyla ilgili personel olduğunu belirtmiş ancak bunu da iş analisti olarak tanımlamıştır.

Katılımcıların verdikleri yanıtlardan kullanılabilirliğin sağlanmasına yönelik organizasyonlarda kullanılabilirlikle ilgili eğitime sahip personel bulunmadığı ve bu görevlerin organizasyonlarda daha çok iş analistleri (K3_İA_1, K2_YG_3, K3_UY_1, K4_KU_1) ya da test mühendisleri (K1_YG_1, K5_TM_1, K6_TM_2) tarafından gerçekleştirildiği görülmektedir. Örneğin K3_İA_1 bu konu hakkında “Kullanılabilirliğe ilişkin bir çalışma gerçekleştirilmiyor. İş analistleri projelerde kullanılabilirliği dikkate almaya çalışarak analiz safhasını yürütüyor” şeklinde görüşlerini belirtmiştir. K2_YG_3 ise “Kullanıcıların ihtiyaçlarını, kullanılabilirliğini belirleyen uygulamaların, kullanıcıların ihtiyaçlarına göre hazırlayan analistler var” şeklinde görüşünü dile getirmiştir. K6_TM_2 da “Product & Testing Departmanı / Product Engineer mühendisleri bu görevi üstlenmektedir.” Şeklinde görüşünü belirtmiştir.

3.2.1.2. Kullanılabilirliğin sağlanmasına verilen önem

Organizasyonlarda uygulanan yazılım projelerinde kullanılabilirliğin sağlanmasına verilen önemi ölçmek adına katılımcılardan “kullanılabilirlik”, “fonksiyonallık”, “görsel tasarım”, “güvenilirlik” gibi kavramları önceliklendirmeleri, bu kavramlardan hangisinin en çok test edildiğini belirtmeleri istenmiştir. Katılımcıların büyük çoğunluğu fonksiyonallık (K1_YG_1, K1_YG_2, K4_KU_1, K3_İA_1, K5_TM_1, K6_TM_2) cevabını vermiştir. Katılımcılardan bazıları fonksiyonallıkla birlikte güvenilirliğin de öncelikli olduğunu (K1_YG_1, K3_İA_1) belirtmişlerdir. Katılımcılardan K4_KU_1 projelerinde fonksiyonallık ve kullanılabilirliğin öncelikli kavramlar olduğunu belirttiği halde, projelerde en çok test edilen, en çok başarının elde edildiği kavramın fonksiyonallık olduğunu belirtmiştir. Yani kullanılabilirlik öncelikli bir kavram olmasına rağmen en çok test edilen ve en başarılı olunan kavram olamamıştır. Katılımcılardan K2_YG_3 de en öncelikli konunun kullanılabilirlik olduğunu söylediği halde fonksiyonallığın en çok test edilen ve başarının üst seviyede elde edildiği kavram olduğunu belirtmiştir. Katılımcılardan K4_KU_1 “Projelerimizde genellikle kullanılabilirlik ve fonksiyonallık daha önceliklidir. En

çok fonksiyonallite test edilir. Proje bitiminde fonksiyonallite konusundaki başarı diğerlerine göre daha yüksektir. BT içerisinde üst seviyelerde konu öncelikli olarak değerlendirilmekte ve tüm organizasyon buna göre çalışmalarını sürdürmektedir” diyerek bu tutarsız durumu şu şekilde örneklendirmiştir. K2_YG_3 ise bu konuyla ilgili “Projelerimizde genellikle kullanılabilirlik daha önceliklidir. En çok fonksiyonallite test edilir. Proje bitiminde fonksiyonallite konusundaki başarı diğerlerine göre daha yüksektir. Önem veriliyor. Kullanıcılarımızı memnun etmek zorundayız. Projeleri sattığımız için de bunlara önem vermek zorundayız. Arka tarafta işi bir şekilde kotarıyoruz ama bunu kullanıcıya yansıtmadan onun istediğini göstermeye çalışıyoruz” şeklinde görüşünü belirtmiştir.

3.2.1.3. Organizasyonda uygulanan yazılım yaşam döngüsü

Organizasyonlarda uygulanan yazılım yaşam döngüsü modeli de hem kurumlar arası hem organizasyon içinde bile çeşitlilik göstermektedir. Kullanılan modeller olarak Çağlayan (K1_YG_1, K3_İA_1, K3_UY_1), V-modeli (K4_KU_1, K5_TM_1) gibi geleneksel yöntemler daha çok belirtilirken, bazı kurumların da projeye göre çağlayan ya da çevik yöntemleri seçtiği (K3_İA_1, K5_TM_1) görülmektedir. K6_TM_2 kendi organizasyonları için geliştirdikleri bir yöntemi kullandıklarını belirtirken, K2_YG_3 belirlenmiş bir yaşam döngüsü kullanmadıklarını da belirtmektedir.

3.2.1.4. Kullanılabilirliğin yaşam döngüsüne entegrasyonu

Kullanılabilirliğin sağlanmasına yönelik aktivitelerin organizasyonlarda mevcut durumda çoğunlukla uygulanmadığı görülmektedir. K3_İA_1 hiçbir aşamada uygulamadıklarını açıklıkla ifade etmektedir. K2_YG_3 kullanılabilirliği yalnızca son aşamada uyguladıklarını, K5_TM_1 ise yine son aşamada hataları bulmak adına uyguladıklarını belirtmektedirler.

Diğer taraftan, katılımcıların hemen hemen hepsi de kullanılabilirliğin sağlanmasına yönelik aktivitelerin yaşam döngüsünün ilk aşamalarından itibaren gerçekleştirilmesi gerektiğini belirtmektedirler (K1_YG_1, K1_YG_2, K4_KU_1, K3_İA_1, K5_TM_1, K3_UY_1). En çok entegre edilmesi gereken aşamalar olarak ön-çalışma, analiz, tasarım ve test adımları belirtilmektedir. K4_KU_1 kullanılabilirliğin en fazla

entegre edilmesi gereken aşamaları analiz ve test olarak belirtirken ayrıca kullanılabilirliği uygulamanın zaman alıcı olduğunu düşündükleri için gerçekleştirmediklerinden de bahsetmektedir. Bu konudaki görüşünü “Kullanılabilirliğin özellikle analiz ve testte olması gerektiğini düşünüyorum bir de ‘Kullanıcı Gözlem testi’ çok zaman gerektirdiği için çoğu zaman atlanıyor. Bunu yapmak isterdik” şeklinde görüşünü belirtmiştir.

3.2.1.5. Uygulanan modelleme yaklaşımları

Organizasyonlarda katılımcıların herhangi bir modelleme yaklaşımı kullanıp kullanmadıkları sorulmuştur. Katılımcılardan çoğunun herhangi bir modelleme yaklaşımı kullanmadıkları öğrenilmiştir. Bunlardan bazıları kullanmadıkları halde bu durumdan dolayı eksiklik hissetmediği (K4_KU_1, K3_İA_1, K5_TM_1, K3_UY_1), K2_YG_3’nin ise eksikliğini hissettiği sonucuna varılmıştır. K2_YG_3 bu durumu “Kullanmıyoruz. Kesinlikle hissediyorum. Projenin okunurluğu yok. Mesela ben projeye sonradan dahil olduğum için eskiden yapılan işleri anlayabilmek için kod okumam gerekti. Bunun yerine şema çizilmiş olsaydı, db diyagramı olsaydı ilişkileri görmek adına daha kolay olurdu. İlişkileri anlayabilmek için tablolara tek tek gidip kendim çiziyorum diyagramları.” şeklinde örneklendirmiştir.

Katılımcılardan K1_YG_1’in ER (Entity Relation Diagram) ‘ı kullanırken, K1_YG_2’nin UML kullandığı öğrenilmiştir. Yine aynı organizasyonun farklı bölümlerinde farklı modelleme yaklaşımlarının uygulanabildiği görülmektedir.

3.2.2. Kullanılabilirlik değerlendirmelerinin gerçekleştirilmesi

Katılımcılara çalıştıkları organizasyonlarda kullanılabilirlik değerlendirmelerini nasıl gerçekleştirdiklerini öğrenmek adına sorular yöneltilmiştir. Alınan yanıtlardan elde edilen bulgular organizasyonlardaki kullanılabilirlik gereksinimlerinin toplanması (KG_2_1), kullanılabilirlik yöntemlerini uygulama (KG_2_2), kullanılabilirlik değerlendirme sonuçları (KG_2_3) konularındadır.

3.2.2.1. Kullanılabilirlik gereksinimlerinin toplanması

Organizasyonlarda katılımcılara kullanılabilirlik gereksinimlerini nasıl topladıkları sorulmuştur. Katılımcıların çoğu gereksinimleri müşterilerden topladıklarını

(K1_YG_1, K1_YG_2, K3_İA_1, K3_UY_1), bir kısmı kullanıcılarla çok yakın temasta olduklarını (K4_KU_1) ve bu görevi analistlerin iş analistlerinin üstlendiğini belirtmiştir (K4_KU_1, K2_YG_3, K5_TM_1). Bu durumu K4_KU_1 şu şekilde örneklendirmiştir; “Bazı standartlarımız var. Kullanıcı ile yakın çalışıyoruz. Bu görevi iş analistleri üstleniyor.”

Organizasyonda bu görevi üstlenen bir kişi (K6_TM_2) daha önceki benzer çalışmaları, internetteki son trendleri ve rakiplerin yaptığı çalışmaları göz önünde bulundurarak gereksinimleri topladıklarını belirtmiştir. Bu durumu şöyle örneklendirmiştir: “Daha önceki benzer çalışmalarımıza, internetteki son trendlere ve rakiplerin yaptığı çalışmalara göz atarak topluyoruz. Product ekibi bu görevi üstleniyor.”

3.2.2.2. Kullanılabilirlik yöntemlerini uygulama

Organizasyonlarda kullanılabilirlik yöntemlerinden hangilerini ve nasıl uyguladıkları sorulmuştur. Katılımcıların çoğu kullanıcı testlerini (K1_YG_1, K4_KU_1, K2_YG_3, K3_UY_1, K6_TM_2), bir kısmı sesli düşünmeyi de (K1_YG_2, K3_UY_1, K6_TM_2) kullandığını belirtmiştir. K4_KU_1 kullanıcı testlerine ek olarak sezgisel testleri (heuristic testing), K6_TM_2 ise stil kılavuzlarını gelecekte kullanmayı planladıklarını söylemiştir. Genellikle kullanıcı testlerini gerçekleştirirken ortalama 5-10 kişinin bu testlerde yer aldığını belirtmişlerdir. Yazılımcı olarak organizasyonda görev alan K2_YG_3 kullandıkları yöntemi anlatırken şu şekilde örneklendirmiştir: “Sadece kullanıcı testleri. Analist test ediyor. Ben test ediyorum daha sonra sahadaki ekibe gönderilip, canlıda test edilmesi sağlanıyor. Sahadaki ekip: Uygulamayı gerçek kullanacak kişiler. Örneğin bizim lojistik projesindeyiz. Müşterinin de kendi e-test ekibi var. O da devreye almadan önce canlıya paralel şekilde kendi de test ediyor. Böylece 5-6 kişi diyebiliriz.”

Teste katılan kişi sayısının proje büyüklüğüne göre değişebileceğini söyleyen K4_KU_1 ise durumu şu şekilde örneklendirmiştir: “Kullanıcı testleri ve sezgisel testleri (heuristic testing) kullanıcılarla gerçekleştiriyoruz. Kişi sayısı proje büyüklüğüne göre değişiyor.”

3.2.2.3. Kullanılabilirlik değerlendirme sonuçları

Katılımcılara organizasyonlarda yapılan kullanılabilirlik değerlendirme sonuçlarının nasıl kullanıldığı sorulmuştur. Katılımcılardan bir kısmı bu sonuçlara müşterilere (K1_YG_1, K2_YG_3), bir kısmı yazılım geliştiricilere (K1_YG_2, K3_UY_1, K6_TM_2) geribildirim olarak sunduklarını söylemişlerdir. K4_KU_1 ise bu sonuçların BT de sorumlu analiz ekiplerince kullanıldığını söylemiştir.

3.2.3. Yazılım geliştirme profesyonellerinin kullanılabilirlik algıları

Katılımcılara çalıştıkları organizasyonlardaki yazılım geliştirme profesyonellerinin kullanılabilirlik algılarını öğrenmek adına sorular yöneltilmiştir. Alınan yanıtlardan elde edilen bulgular daha çok kullanılabilirlik değerlendirmesinin yazılım geliştirme sürecinde uygulanmasının avantajları ve dezavantajlar (KG_3_1), kullanılan yazılım yaşam döngüsüne kullanılabilirlik entegrasyonu (KG_3_2), Yazılım yaşam döngüsüne kullanılabilirlik entegrasyonunda uygulanabilecek aktiviteler (KG_3_3) konularındadır.

3.2.3.1. Kullanılabilirlik avantajları ve dezavantajları

Organizasyonlarda katılımcılara yapılan kullanılabilirlik değerlendirmesinin yazılım geliştirme sürecinde uygulanmasının avantaj ve dezavantajları sorulmuştur. Genellikle dezavantaj olarak çoğu katılımcı zaman kaybı (K1_YG_1, K1_YG_2, K3_İA_1, K2_YG_3, K6_TM_2), parasal kayıp (K1_YG_1) ve daha fazla araştırma ile hazırlanma (analiz, dokümantasyon vb.) gerektirmesi (K6_TM_2) gibi sebepler öne sürülmüştür. Avantaj olarak ise müşteri memnuniyeti sağlaması (K1_YG_2, K2_YG_3), ürünün daha çok satma potansiyeli oluşturması (K1_YG_2), kolay anlaşılır fonksiyonalite, gereksiz geliştirmeden kurtulma, daha az hata ve daha az bakım maliyeti, daha kolay otomasyon (K6_TM_2), kullanıcıya daha yakın önyüzler tasarlanması (K4_KU_1) gibi sebepler gösterilmiştir. K2_YG_3 ise dezavantajının olduğunu düşünmediğini, yazılım sürecinde uygulanmasının müşterilerinin beklentilerini daha iyi karşılayacağını düşündüğünü belirtmiştir. K6_TM_2 kullanılabilirlik değerlendirmesinin yazılım geliştirme sürecinde uygulanmasının avantajları ve dezavantajlar konusunda yorumlarını şu şekilde örneklendirmiştir: “Avantajları: Kolay anlaşılır fonksiyonalite, gereksiz geliştirmeden kurtulma, daha

az bug ve daha az bakım maliyeti, daha kolay otomasyon. Dezavantajlar: Daha fazla araştırma ve hazırlanma (analiz, dokümantasyon vb.) süreci gerektiriyor. Dolayısıyla zaman ve yetkin personel gerektiriyor.”

3.2.3.2. Yazılım döngüsüne kullanılabilirlik entegrasyonu

Katılımcılara organizasyonlarda kullanılan yazılım yaşam döngüsüne kullanılabilirliğin entegrasyonun ne derecede olduğu sorulmuştur. Katılımcılardan çoğu kullandıkları yazılım yaşam döngüsüne kullanılabilirlik entegrasyonun olmadığını söylemiştir (K1_YG_1, K1_YG_2, K3_İA_1, K2_YG_3, K5_TM_1, K3_UY_1). K1_YG_2 konu hakkındaki görüşünü “Tam olarak entegre olmadığını düşünüyorum. Sebebinin gerekli önemin verilmemesi ve sistemli bir tasarım süreci olmamasına bağlıyorum” diyerek örneklendirmiştir. K2_YG_3 ise “Entegre olduğunu düşünmüyorum. Sadece son aşamalarda testlerle” şeklinde örneklendirmiştir.

3.2.3.3. Kullanılabilirlik entegrasyonunda aktiviteler

Katılımcılara organizasyonlarında uyguladıkları yazılım yaşam döngüsüne kullanılabilirliğin entegrasyonunda hangi aktivitelerin uygulanabileceği sorulmuştur. Katılımcılardan K1_YG_2, K4_KU_1, K5_TM_1 bunun eğitim ile mümkün olabileceğine değinmişlerdir. K6_TM_2 personel yetkinliğinin artırılması, stil kılavuzlarının kullanılması ve otomasyon testleri gibi aktivitelerin uygulanmasını önermiştir. K4_KU_1 bu konuda hakkında “Analiz ve test fazlarına uygun kontrol noktalarını ekleyerek. Analiz ve yazılım ekiplerini eğiterek” şeklinde görüşünü belirtmiştir. K3_İA_1 ise konuyla ilgili “Projeye kullanılabilirlik konusunda uzman personellerin dahil edilmesi ile birlikte zaman kaybına yol açmadan kullanılabilirlik süreçleri entegre edilebilir” şeklinde görüşünü belirtmiştir. K5_TM_1 ise konu hakkında “O bilincin olması gerekiyor yöneticilerde. Eğitim şart. Kullanılabilirlik eğitimleri verilebilir önemini kavratmak için” şeklinde görüşünü dile getirmiştir. K6_TM_2 ise “Son kullanıcıya yapılan testlerin artırılması, personel yetkinliğinin geliştirilmesi ve stil kılavuzlarının hazırlanıp otomasyon testleri ile desteklendikçe daha etkili olacaktır” şeklinde görüşünü belirtmiştir.

3.2.4. Görüşmelerin Değerlendirilmesi

Yapılan görüşmeler ile organizasyonların kullanılabilirlik konusunda tecrübeleri, bakış açıları, algıları değerlendirilmiştir. Bu değerlendirmeler sonrasında önerilecek modelin oluşumuna katkı sağlayacak bilgiler çıkarılmıştır.

Organizasyonların çoğunda İBE alanında çalışan uzman personel eksikliğinin olduğu ve bu eksikliğin genellikle analistlerce ve test mühendisleriyle giderildiği görülmüştür. Katılımcılardan organizasyonlarda proje geliştirimi sırasında verilen güvenilirlik, fonksiyonalite, kullanılabilirlik, görsellik gibi kavramların öncelik sıralamasının yapmaları istendiğinde fonksiyonalite kavramının çoğu organizasyon için öncelikli olduğu görülmüştür. Bazı katılımcılar kullanılabilirliğin de fonksiyonalite kadar öncelikli olduğunu söylese de en çok test edilen kavramın fonksiyonalite olduğunu söyleyerek çelişkili bir durumu ortaya koymuşlardır. Mevcut organizasyonların uyguladıkları yazılım yaşam döngüsüne kullanılabilirliği ne kadar entegre ettiklerine bakıldığında kullanılabilirliğin sadece yazılım yaşam döngüsünün son aşamalarında uygulanan kullanılabilirlik testi ile sağlanmaya çalışıldığı görülmüştür. Bir katılımcı da son aşamadan çok önce, analiz fazında da entegre edilmesi gereken bir konu olduğundan bahsetmiştir.

Organizasyonların çoğunun yazılım projelerindeki gereksinimleri kullanıcılarla sürekli iletişim halinde kalarak topladıkları gözlenmiştir. Bu görevin genellikle analistlerle gerçekleştiğini de belirtmişlerdir.

Organizasyonların uyguladıkları kullanılabilirlik yöntemleri ne bakıldığında çoğu organizasyonun en çok kullanıcı testlerini kullandıkları belirlenmiştir. Sadece bir organizasyonun stil kılavuzları ile çalışma planı yaptığı, bir organizasyonun da sezgisel değerlendirme yönetimini uyguladığı öğrenilmiştir.

Organizasyonlara kullanılabilirliği yazılım yaşam döngüsüne entegre etmenin avantajlı olduğuna değinmiştir. Ancak diğer taraftan genellikle katılımcıların hepsi kullanılabilirliğin organizasyonlar için zaman kaybı olarak görüldüğünün de bilgisini vermiştir. Bundan dolayı kullanılabilirlik hala şirketlerce tam benimsenememiştir.

Yazılım yaşam döngüsüne kullanılabilirliğin entegrasyonuna uygulanabilecek aktiviteler sorulduğunda analist fazında, yani daha döngünün erken fazında

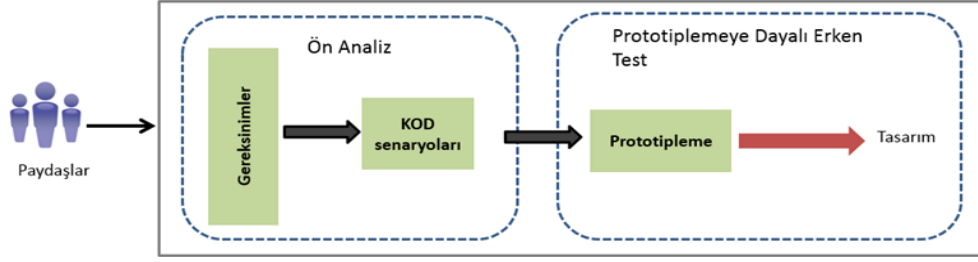
uygulanmaya başlanması gerektiği üzerinde durulmuştur. Bir kullanıcı da bu konuda personel yetkinliğinin artırılması, stil kılavuzlarının kullanılarak otomasyon testleriyle desteklenmesi gerektiği gibi bir öneride bulunmuştur.

3.3. Modelin İlk Sürümü

Bu bölümde önerilen KOD yazılım geliştirme modelinin ana bileşenleri anlatılmaktadır ve açıklanmakta olan önerilen modelin ilk sürümünden bahsedilirken kısaca KOD v.1 olarak isimlendirilecektir. Modelin birinci sürümü yapılan alanyazın araştırması ve gerçekleştirilen anket çalışması ve yapılan görüşmelerden elde edilen bulgular doğrultusunda geliştirilmiştir. Alanyazından yazılım mühendisliği alanından özellikle çevik metodolojilerin prensipleri adapte edilmiş, İBE alanından ise kullanıcı merkezli tasarımının iyi pratikleri entegre edilmiştir. Çevik metodolojilerden özellikle Davranış Odaklı Geliştirme (DOG) temel alınmıştır ve DOG yaklaşımı kullanılarak yazılan senaryolar, önerilen modelde KOD senaryoları olarak isimlendirilmiştir.

Önerilen modelde kullanıcının ihtiyaçlarını doğru anlamak öncelikli olup, onların taleplerini karşılayan ürünü ortaya çıkartmak için kullanıcıları her aşamada merkezde tutan bir model sunmak amaçlanmıştır. Çevik yöntemlerde uygulanan analiz ve tasarım, geliştirme, test, değerlendirme ve dağıtım adımları bu modelde de sürecin ana basamaklarını oluşturmaktadır. Ancak bu ana basamaklara bazı alt fazların dahil edilmesiyle kullanıcıların her aşamada sisteme katkıda bulunmasının sağlanması hedeflenmektedir. KOD v.1' in ana fazları Analiz ve Tasarım, Gerçekleştirme, Değerlendirme ve Dağıtımdır. Analiz ve Tasarım fazı kendi içinde Ön Analiz ve Erken Test aşaması alt fazlarından oluşmaktadır. Gerçekleştirme fazında KOD senaryolarına dayalı arayüz ve sınıf testleriyle geliştirme yapılması önerilmektedir. Sınıf testi ve arayüz testi yazımı gerçekleştirme fazının önerilen metotlarından.

Şekil 3.4'te gösterildiği üzere her yinelemede dağıtım dışındaki tüm fazlar uygulanmaktadır. Dağıtım tüm yinelemeler sona erdiğinde müşteriye teslim edilebilecek versiyonun sunulduğu fazdır. Yinelemeli yaklaşım uygulanırken Çevik yaklaşımlardan olan Uyarlanabilir Yazılım Geliştirme (Adaptive Software Development-ASD) baz alınmıştır.



Şekil 3.5. Analiz ve tasarım fazı

Şekil 3.5’te de anlaşılacağı üzere gereksinimlerin KOD senaryolarına dönüştürüldüğü alt faz ön analiz, prototipleme ile oluşturulan tasarımların test edilebildiği alt faz erken test olarak adlandırılmaktadır. Ön analiz aşamasının detayları 3.3.1.1, prototiplemeye dayalı erken test aşaması da 3.3.1.2 başlığı altında anlatılmaktadır.

3.3.1.1. Ön analiz aşaması (pre-analysis)

KOD v.1’in analiz & tasarım fazının alt fazlarından ön analiz aşamasında, sistemin ihtiyaçlarının gerçeği yansıtması adına gerçek kullanıcılarla ve müşterilerle irtibata geçilerek toplanan gereksinimler kullanılmaktadır. Şekil 3.5’te görüldüğü üzere paydaşların sistemle ilgili topladıkları gereksinimler analistlerin yardımıyla DOG(BDD) formatına dönüştürülerek KOD senaryoları şeklinde tanımlanmaktadır. Bu aşamada analistlerin müşteri ve kullanıcıların sürekli iletişim halinde olması gerekmektedir.

3.3.1.2. Prototiplemeye dayalı erken test

KOD v.1’in analiz ve tasarım fazının alt fazlarından prototiplemeye dayalı erken test, sistemin gereksinimlerinin tasarıma dönüştürüldüğü ve tasarlanan ekranlar üzerinden erken testlerin yapıldığı aşamadır. Oluşturulan KOD senaryoları her gereksinimin ayrıntılı anlatımı olduğundan ekranların oluşturulmasını kolaylaştırmaktadır. Ekran tasarımları için kâğıt prototipleme ya da çevrimiçi yazılım araçları kullanılabilir. Kullanılabilirlik testleri hazırlanan prototipler (kâğıt veya yazılım prototipleri) üzerinden yapılabilir. Kullanılabilirlik testleri yöntemin uygulanması hakkında yeterli bilgiye sahip kişiler tarafından gerçekleştirilebilir. (URL-11, 1997). Tipik bir kullanılabilirlik testinde izlenmesi gereken dört adımı şu şekilde sıralamıştır:

- Testin planlanması: Program ve görevlerin geliştirilmesi. Katılımcıların bulunması
- Kişisel oturumlar: Katılımcıların teste dahil olması.
- Gözlemciler: Katılımcı hareketlerinin gözlemlenmesi.
- Kullanılabilirlik raporu: Gözlemlerin sonuçlarının incelenmesi (URL-7, 2009)

Kullanılabilirlik testinde kullanılacak görevler için KOD senaryoları kullanılmaktadır. Böylece görev hazırlamak için de zaman kaybedilmemektedir. KOD v.1’de kullanılabilirlik testleri gereksinimleri sağlayan gerçek kullanıcılar ve müşteriler ile gerçekleştirilir. Sistemin ilk aşamalarında kullanılabilirlik testlerinin uygulanmasıyla, tasarımların başarısı erken safhalarda dahi ölçülebilmektedir.

Erken testlerde, müşterilerle prototipler üzerinden sistemin test edilmesi sırasında Wizard Of Oz (Kelley, 1983; Green ve Wei-Haas, 1985) tekniğinin kullanılması gerekebilir. Wizard of Oz Tekniği henüz tamamlanmamış bir sistemin test edilmesini sağlayan tekniklerden biridir. Bu teknikte kullanıcı testleri gerçekleştirdiği sırada, tamamlanmamış fonksiyonlar deneyi gerçekleştiren kişi ya da kişilerce simüle edilmektedir (Kelley, 1983; Green ve Wei-Haas, 1985)

3.3.2. Gerçekleştirme

Bu fazda gereksinimler yazılım ekibi tarafından geliştirilmektedir. DOG (BDD) yaklaşımı KOD senaryoları ile KOD v.1’de sadece analiz fazında değil, gerçekleştirme fazında da kullanılmaktadır. Yazılımı kolayca yapılandırmak, kusursuz kodlamayı etkinleştirmek için analiz sırasında kullanılan KOD senaryoları bu fazda da kullanılmakta ve bu aşamada sınıf testlerinin ve arayüz testlerinin yazılmasına odaklanılmaktadır. Yazılım ekibine ulaşan KOD senaryoları bazı yazılım araçları kullanılarak doğrudan test fonksiyonlarına dönüştürülebilmektedir. KOD senaryolarında olduğu gibi sınıf testleri için Specflow, WatiN vb. kullanıcı arayüzü testleri için Mocha, Jasmine vb. frameworkler kullanılarak, hazır test metotlarının yaratılması sağlanmaktadır (URL-2, 2015; URL-5, 2014)

3.3.3. Değerlendirme

Değerlendirme fazı modelin merkezinde yer almaktadır ve entegre edilmiş gereksinimlerle beraber sistemin kullanıcı deneyimi açısından başarısı değerlendirilmektedir. Yazılım yaşam döngüsünde karar mekanizması oluşturarak bir sonraki adımın hangi faza dahil edileceğinin kararı burada verilmektedir.

Geliştirilen ürünün, kullanıcı deneyimini sağlayabilmesi için kullanılabilirlik testi yapılmaktadır. Yapılan kullanılabilirlik testindeki başarıya/başarısızlığa göre bir sonraki faza karar verilmektedir. Eğer ürün kullanıcı için kolay kullanılabilir, memnuniyet verici değilse, analiz fazına tekrar geri dönülmektedir.

KOD v.1'in değerlendirme fazında sisteme kullanılabilirlik teknik ve yöntemlerinden sistemde yer alan paydaşlarca tercih edilenler uygulanabilmektedir. Ancak önerilen model KOD v.1'de yapılan alanyazın araştırmaları, anket ve görüşme sonuçlarına göre organizasyonlarca da tercih edilen yöntemlerden olan sezgisel (heuristic) değerlendirme, stil kılavuzları ya da kullanıcı testi kullanılması önerilmektedir.

3.3.4. Dağıtım

Dağıtım, gerçekleştirilen yinelemelerin tamamlanması sonrasında müşteri ortamına koyulmak istenen kodların (entegre edilmiş senaryoların) paketlenme işlemidir. Bu aşama sonrasında tutarlı sürümlerin kullanıcılara ulaştırılması en öncelikli amaçlardandır. Dağıtım fazından önce son kez kullanıcı kabul testleri yapılarak sistemin kullanıcılar tarafından test edilmesi sağlanmaktadır. Böylece paydaşların (müşteriler, kullanıcılar vb.) karşılaşılabileceği sorunlar minimuma indirilmektedir.

3.3.5. Yinelemeler ve UYG

KOD Yazılım Geliştirme Modeli'nin önemli özelliklerinden biri de yinelemelerle oluşturulmuş olmasıdır. Yinelemeler, UYG (ASD) modelinin zaman kısıtlı yaklaşımı örnek alınarak oluşturulmuştur. Her yinelemede uygulanan geliştirme adımı süreleri eşit değildir ve birbirine paralel olarak ilerlemektedir. Analiz ve tasarım, gerçekleştirme, değerlendirme adımları yinelemelerle tekrarlanırken, dağıtım adımı birkaç yinelemeden sonra uygulanmaktadır. Çünkü dağıtım müşteriye çıkacak sürümün çıkarıldığı geliştirme adımını temsil etmektedir.

3.4. İhtiyaç Modelinin Değerlendirilmesi ve Bulguları

KOD v.1 modeli bir yüksek lisans dersi kapsamında öğrenciler tarafından ders projelerini geliştirmeleri sırasında uygulanmıştır. Bu uygulama sonrasında modelin değerlendirilmesine yönelik olarak öğrenciler ile gerçekleştirilen görüşmelere ait bulgular bu bölümde paylaşılmaktadır.

3.4.1. Kullanıcı deneyimi konusunda farkındalık

Katılımcılara çalıştıkları organizasyonlar görev aldıkları pozisyonlar ile ilişkili olarak ya da varsa aldıkları eğitimlerle kullanıcı deneyimi konusunda kazandıkları tecrübelerle ilgili sorular sorulmuştur. PG1_T2, PG1_T3, PG2_T1 bu konu hakkında tecrübelerinin olmadıklarını söylemişlerdir. PG1_T1 ise daha önce PMI, PMBOK, PMP projelerinde yer aldığını söyleyerek kullanıcı merkezliliğin öneminin farkında olduğunu vurgulamıştır. PG1_T1 bu durumu şöyle örneklendirmiştir: “Kullanıcı en önemli faktör. Paydaşların yorumunu almak çok önemli. Son çalıştığım firma uluslararasıydı ve yabancı ortaklar vardı. Ülkelerine göre süreçleri değişebilir. Kişilerin psikolojik durumları çok önemlidir. Daha önce PMI, PMBOK, PMP projelerinde yer aldım. RACI: Görev atama matrisi kullanılmaktaydı.”

3.4.2. Çevik yöntemler konusunda tecrübeler

Katılımcılara çalıştıkları organizasyonlarda ya da varsa aldıkları eğitimlerle çevik yöntemler konusunda kazandıkları tecrübeler hakkında sorular sorulmuştur. Alınan yanıtlardan elde edilen bulgular daha çok kullanılan çevik yöntemler (KG_2_1), çevik yöntemlerden TOG/DOG (TDD/BDD) hakkında tecrübe konularındadır.

3.4.2.1. Kullanılan çevik yöntemler

Katılımcılara daha önce çevik yöntemler ile ilgili tecrübelerinin olup olmadığı, hangi çevik yöntemleri kullandıkları sorulmuştur. PG1_T1 ve PG2_T1 daha önce çevik yöntemlerden SCRUM kullandıklarını dile getirmişlerdir. Diğer katılımcılar ise çevik yöntemlerle ilgili tecrübelerinin olmadıklarını söylemişlerdir.

PG2_T1 kendi durumunu şu şekilde örnelemiştir: “Evet, tecrübem vardı. Scrum kullandım. Uygularken yapılan aktiviteler: Sprint Meeting, Planning, Daily Review Meeting, Sprint Review Meeting.”

3.4.2.2. Çevik yaklaşımlardan TOG/DOG (TDD/BDD) hakkında tecrübe

Katılımcılara çevik yaklaşımlardan TOG/DOG (TDD/BDD) hakkında tecrübelerinin olup olmadığı sorulduğunda sadece PG2_T1 bilgisi olduğunu; ancak herhangi bir projede kullanmadığını söylemiştir. Diğer katılımcılar da ilk defa ders projesinde tecrübe sahibi olduklarını dile getirmişlerdir. PG1_T1 konu hakkında durumunu şu şekilde örnelemiştir: “İlk defa ders projesinde kullandım. Müşterinin istediğiyle yazılımcıların görebildikleri birbirinden farklıdır. Müşteriyi işin içine girer girmez çekmek zorundasınız. DOG (BDD) buna göre çok yazılımsal kalabilir. Kullanıcı DOG (BDD) la direk muhatap olursa, anlayamadığı bir şeyi yorumlayamayabilir.”

3.4.3. KOD analiz ve tasarım safhası

Katılımcılara önerilen modelin analiz ve tasarım safhasında uyguladıkları yöntemler hakkında sorular sorulmuştur. Alınan yanıtlardan elde edilen bulgular daha çok ders projesinde gerçekleştirilen süreç ve takım çalışması (KG_3_1), DOG (BDD)’nin sürece katkısı (KG_3_2) konularındadır.

3.4.3.1. Ders projesinde gerçekleştirilen süreç ve takım çalışması

Bu kategoride katılımcılar ders projesinde gerçekleştirilen süreçleri ve takım çalışmalarını anlatmışlardır. Önerilen modelin analiz ve tasarım safhasını gerçekleştiren katılımcılardan PG1_T1 yorumlarını şu şekilde örneklendirmiştir: “İlk önce farklı personalardan gereksinimleri topladık. Toplanan gereksinimleri KOD senaryolarına dönüştürdük. Senaryolara göre ekran tasarımlarını yapıp, kullanıcı testlerini gerçekleştirdik.”. PG1_T2 ise şu şekilde yorum yapmıştır: “Farklı kullanıcı profilleri oluşturduk. Gereksinimleri topladık. Gereksinimleri KOD senaryolarına dönüştürdük. Ekranları tasarımlarını ve testlerini yaptık. İnsan kaynakları projesi daha görsel iken, finansal bir proje olduğundan daha görsellikten uzak. Kafada canlandırılabilmesi, soyutlaşması adına önce BDD sonra prototipleme olması yerine bazı durumlarda prototiplemeyi öne almayı öneriyorum.”. PG1_T3’ün yorumu şu şekildedir: “Önerilen modelin analiz ve tasarım safhasını gerçekleştirdik.

Gereksinimlerin toplanıp KOD senaryolarına dönüştürülmesi ve ekran tasarımlarını yaratıp kullanıcı testlerini yaptık.”. PG2_T1 ise şu şekilde yorum yapmıştır: “Önce gereksinimleri topladık ve kullanıcı_personaların oluşturduk. Sonra kullanıcı senaryolarını yazdık cucumber kullanarak. Ve kullanıcı senaryosuna göre prototip oluşturduk. Sonunda prototip üzerinden kullanıcı testi yaptık.”.

3.4.3.2. DOG (BDD)’un sürece katkısı

Katılımcılara kullanılan DOG (BDD) yaklaşımının sürece olan katkısı sorulmuştur. PG1_T1 senaryolarının yazımının kolay olduğunu, senaryolar sayesinde tüm durumların ele alındığını söylemiştir. PG1_T2 ve PG2_T1 ise DOG (BDD) sayesinde prototiplerin daha kolay oluşturulduğunu dile getirmiştir. PG1_T3 DOG (BDD) sayesinde sistemin canlı dokümantasyonunun sağlandığını belirtmiştir. PG1_T1 bu konu hakkındaki yorumlarını şu şekilde örneklendirmiştir: “Kolay bir dildi. Çıkarılan personaların KOD yazımına katkısı oldu. Personalar oluşturulurken caseler ele alınıyor. Senaryolar yazılırken de tüm caseler düşünülmüş oluyor.”. PG1_T2 ise şu şekilde yorum yapmıştır: “Olayların sırası DOG (BDD) ile önceden düşünüldüğünden prototiplemeler(ekranlar) daha çabuk oluşturuldu.”. PG1_T3’ ün yorumları şu şekildedir: “KOD senaryoları aslında projenin canlı dokümantasyonunu da sağlayan bir yaklaşım diyebiliriz. Geliştirilmek istenen, sistemde yer alacak her gereksinim ayrıntılı şekilde KOD senaryolarıyla dokümente edilmektedir.”.

3.4.4. DOG (BDD) ve kullanıcı odaklı tasarım

Katılımcılara DOG (BDD) yaklaşımının kullanıcı odaklı tasarıma katkısı sorulmuştur. PG1_T1 paydaşlar arasında ortak dil kullanımı sağladığının üzerinde durmuştur. PG1_T2 ise her durumun senaryolar sayesinde gözden geçirilmesiyle yanlış yapma riskini azalttığı konusuna değinmiştir. PG1_T3 de senaryolar sayesinde kullanıcı rollerinin iyi kavrandığının üzerinde durmuştur. PG2_T1 senaryolarla kullanıcıların sistemde neyi nasıl yapmak istediklerini belirtebildiklerini söylemiştir. PG1_T1 konu ile ilgili yorumlarını şu şekilde belirtmiştir: “Paydaşlar arasında ortak dille konuşulmasını sağlıyor.”. PG1_T2 ise şu şekilde yorum yapmıştır: “Her senaryo ayrıntılı olarak düşünüldüğünden ve senaryolara döküldüğünden yanlış yapma olasılığı azalıyor ve kullanıcının hangi adımda ne yapacağı önceden kestirilebiliyor.”. PG1_T3 konu ile ilgili şu ifadeleri paylaşmıştır: “KOD senaryolarını oluştururken

kullanıcı hikâyeleri_üzerinde durduk. Aslında her kullanıcı hikâyesi_senaryoların özet bilgisiydi ve kullanıcı bilgisini de veriyordu. Böylece roller iyice kavranmış oldu. Roller daha iyi kafada canlandığından kendimizi onun yerine koyup senaryoları çoğaltabildik. Ve KOD senaryoları kullanılabilirlik testinde kullandığımız senaryolar oldu.” PG2_T1’in konu ile ilgili görüşleri şu şekildedir: “Behavior Driven Development kendi adından da belli olduğu gibi kullanıcının sistemde neyin nasıl yapmak istediğini belirttiği için. Senaryo ve featurelar da ona göre belli olur.”

3.4.5. Görüşmelerin değerlendirilmesi

Yüksek lisans öğrencileriyle gerçekleştirilen modelin değerlendirme çalışması sonrasında görüşmeler yapılmıştır. Bu görüşmeler sonrasında modelin iyileştirilmesi sırasında kullanılabilecek önemli bilgiler elde edilmiştir.

Katılımcılar önerilen modelin analiz ve tasarım aşamasını uygulamışlardır. Bu uygulamayı gerçekleştirmek için önce farklı personalar tanımlamışlardır. Tanımladıkları personalara göre gereksinimleri toplamış, sonrasında bu gereksinimleri KOD senaryolarına dönüştürülmüştür. KOD senaryoları yazılırken bir yandan da ekran tasarımları geliştirilmiştir. Son aşama olarak da tasarlanan ekranlar kullanılabilirlik açısından teste tabi tutularak değerlendirilmiştir. Katılımcıların çoğu KOD senaryoları yazımının kolay olduğunu ve bu senaryoların yazımı sırasında proje ile ilgili tüm durumların ayrıntılarıyla düşünülebildiğinden bahsetmişlerdir. Böylece prototiplemenin daha kolaylaştığını da vurgulamışlardır. Bununla birlikte yazılan KOD senaryoları ile projenin canlı dokümantasyonunun da sağlandığı söylenmiştir.

Çıkarılan sonuçlara göre önerilen KOD Yazılım geliştirme modelinin analiz ve tasarım aşamasının başarılı bir şekilde uygulandığı söylenebilmektedir. DOG (BDD) çevik yaklaşımının kullanım kolaylığı ve projede paydaşlarca kolay anlaşılır yapısıyla senaryoların kolayca yazılmasına ve böylece ekranların tasarlanmasına olumlu katkıda bulunmuştur.

3.5. İyileştirmeler

Kullanıcı Deneyimi dersi projesi sonrasında öğrencilerle görüşülmüş ve onlardan alınan geri bildirimlere göre modelde bazı değişikliklere gidilmiştir. Öğrencilerin bir

kısmı KOD senaryolarının yazılırken, gözden kaçırılan gereksinimlerin tespit edilebilmesi ve tasarımın somutlaştırılması için eş zamanlı olarak prototiplemeyi de kullandıklarını ifade etmişlerdir. Bu geribildirim sonrasında sıralamalı olarak ilerleyen KOD senaryoları ve prototipleme birbirini besleyen paralel süreçlere dönüştürülmüştür.

Kullanıcı ve müşterilerden toplanan gereksinimlerin KOD senaryoları ile olması düşünülürken, öğrencilerden gelen ortak geribildirimler üzerine gereksinimlerin direk KOD senaryosu formatında alınmasından vazgeçilmiştir. Toplanan gereksinimlerin proje grubundaki analistler ve yazılımcılar tarafından KOD senaryolarına dönüştürülmesi mantıklı bulunmuş, herhangi yanlış anlamaya mahal vermemek ve her süreci kontrol altında tutmak adına müşterilerin de yer alması sağlanarak sürecin ilerlemesine karar verilmiştir.

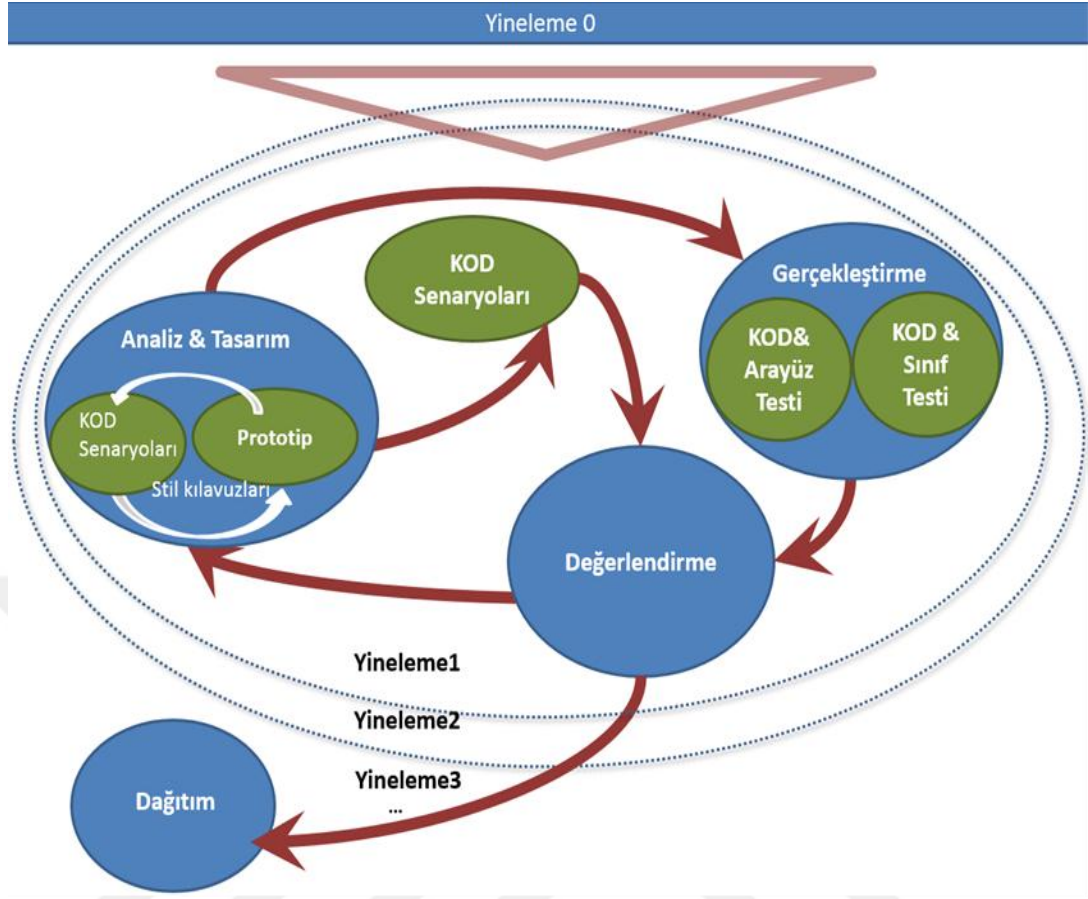
4. KOD YAZILIM GELİŞTİRME MODELİ'NİN SON SÜRÜMÜ

Kullanıcı Deneyimi Tasarım Dersi Projesi'nde modeli uygulayan öğrencilerden gelen geri bildirimler ve alanyazın taramaları sonrasında KOD (Kullanıcı Odaklı Davranış) Yazılım Geliştirme Modeli son sürümü geliştirilmiştir. Raporda önerilen modelin ikinci fazı için yalnızca KOD kısaltması kullanılacaktır. Çevik metodolojinin prensiplerinin adapte edilmesi, kullanıcı odaklı tasarımın iyi pratiklerinin entegre edilmesiyle kullanıcının merkezde olduğu bir modelin oluşturulması amaçlanmıştır.

Çevik prensiplerden özellikle DOG (BDD) yaklaşımı modelin temelini oluşturmaktadır. Sürecin başlangıç fazından itibaren müşteri ve kullanıcılarla olan ilişkiler ön planda tutularak, gereksinimlerin projeye yansıtılabilmesi hedeflerden biri olmuştur. Müşterilerle, yazılım, test ve analiz ekibi arasında ortak dilin konuşulmasını sağlayan DOG (BDD) yaklaşımı kullanılarak sistemin gereksinimlerini detaylandıran senaryolar oluşturulmaktadır.

KOD yazılım geliştirme modeli, Şekil 4.1'de gösterildiği üzere temel olarak analiz ve tasarım, gerçekleştirme, değerlendirme ve dağıtım fazlarından oluşmaktadır. Bu fazların en başında Yineleme 0 adımı bir kez uygulanmaktadır. Yineleme 0 fazındaki amaç gereksinimleri sınıflandırmak ve stil kılavuzları oluşturmaktır.

Modelde, yinelemeli bir yaklaşım kullanılarak, öngörülebilirliği en üst seviyeye çıkarmak ve riski kontrol etmek amaçlanmaktadır. Çünkü her yineleme bir önceki aşamanın kontrolünü de yaparak döngünün devamlılığını sağlamaktadır. Yinelemelerin süreleri UYG (ASD) metodu referans alınarak belirlenmiştir. Böylece müşterinin gereksinimlerini karşılayan ürün çıkana kadar kendini besleyen bir sistem tasarlanmıştır.



Şekil 4.1. KOD Yazılım Geliştirme Modeli Son Sürüm Üst Seviye Gösterimi

4.1. KOD Yazılım Geliştirme ve KOD Senaryosu

DOG (BDD) yaklaşımının temelinde yer alan TOG (TDD), kısa geliştirme döngülerine odaklanan evrimsel bir geliştirme yaklaşımıdır. Adından da anlaşıldığı üzere herhangi bir fonksiyonel kod yazmadan önce otomatik testler yazmaya dayanır (URL-1, 2014).

DOG (BDD), TOG (TDD) ve KTOG (ATDD) nin en iyi uygulamalarını bütünleştirmektedir. DOG (BDD) yaklaşımında gereksinimler kullanıcı hikâyeleri ile özetlenerek, senaryolarla detaylandırılmaktadır. Önerilen modelde kullanıcı gereksinimleri kullanıcı odaklı davranış (KOD) senaryoları halinde KOD senaryo formatında olduğu şekilde tanımlanmaktadır. Bu senaryolar Tablo 4.1’de gösterildiği gibi belirli bir dille yazılmaktadır. Böylece tüm paydaşlar tarafından kolayca anlaşılabilir.

Tablo 4.1. KOD senaryo formatı örneği

Scenario: Ad		Hesaptan para çekme
Given	Durum	Given: Bakiyesi olan bir hesap
When	Olay	When: Müşteri para çekmek istediği zaman
Then	Çıktı	Then: Paranın hesaptan düştüğüne emin ol

KOD yazılım geliştirme modelinin birçok fazında KOD senaryoları Tablo 4.2’de özetlendiği şekli ile farklı amaçlarla kullanılmaktadır.

Tablo 4.2. DOG (BDD)'nin KOD senaryoları kullanımı ile KOD Yazılım Geliştirme Modeli’nde kullanım yeri ve amaçları

Faz	Amaç	Girdi	Çıktı
Analiz ve tasarım	Toplanan gereksinimlerin müşteriler ve kullanıcılar tarafından her yinelemede KOD senaryosu olarak tanımlanması	Toplanan gereksinimler	- KOD senaryoları -Güncel proje dokümanı(KOD senaryoları kolay okunabilir yapısıyla doküman formatına kolayca dönüştürülebilir.)
Gerçekleştirme	-Analiz ve tasarımdan gelen KOD senaryolarının kod yazımında kullanılmasıyla geçerlemelerinin yapılması, -Aynı senaryolarla otomatize edilmiş testlerin yazılarak, otomatik olarak kodların çalıştırılması, kullanıcı arayüz testlerinin çalıştırılması	KOD Senaryoları	Geçerli kod otomatize kullanıcı arayüzü testleri
Değerlendirme	-İlk fazda müşteriler ve kullanıcılarca hazırlanan KOD senaryolarının kullanılabilirlik testinde kullanılması -Her iterasyonda kullanıcılara sunulabilecek güncellenmiş KOD senaryolarının test edilebilmesi	KOD Senaryoları	Kullanılabilirlik testinden geçmiş kod

Aşağıda KOD senaryolarının kullanımı ile eksiklerin, hataların zamanında tespit edilebilmesine dayalı olarak müşteri, kullanıcı ve yazılımcıların lehine sonuçlanan durumlar örneklendirilmiştir.

Tablo 4.3. Modelde müşteriye koruma altına alan durum incelemeleri

	Analiz ve tasarım KOD senaryoları	Gerçekleştirme fazında	Değerlendirme fazında	Sonuç	Yorum
Durum 1	A, B, C, D	A, C senaryoları sınıf testi sonrasında başarısız olurken, B ve D başarılı olmaktadır.	-	A ve C senaryoları entegre edilmemiş olabilir. A ve C senaryoları hedeflendiği gibi çalışmayabilir.	Gerçekleştirme basamağında eksikler tespit edilebilmektedir. Daha kullanıcıya ulaşmadan, eksik ya da hatalı senaryo entegrasyonun düzeltilmesi sağlanabilir.
Durum 2	A, B, C, D	A, B, C, D sınıf testi sonrasında başarılı	A ve D senaryoları kullanılabilirlik testi ile test edildiğinde başarısız	A ve D senaryoları kullanılabilir olmayabilir. İstenildiği gibi sisteme entegre edilmemiş olabilir.	

Tablo 4.3’de anlatıldığı üzere gereksinimlerin ifade edildiği A, B, C, D KOD senaryolarından bazılarının gerçekleştirme fazında yer alan sınıf testlerinden geçemediğini varsayarsak, sınıf testleri sayesinde hangi gereksinimlerin sistemde yer almadığı değerlendirme fazında önce belirlenmiş olmaktadır. Diğer anlatılan durumda ise sınıf testlerinden geçen ancak değerlendirme fazındaki kullanılabilirlik testinden geçemeyen A ve D KOD senaryolarının olduğunu varsayarsak, daha kullanıcıya ulaşmadan sisteme entegre edilmemiş ya da kullanıcı dostu olmayan entegrasyonların varlığı tespit edilmiş olmaktadır.

Örnek olarak Yineleme 1’de A, B, C KOD senaryoları, Yineleme 2’de D, E, F KOD senaryolarının, yineleme 3’te de G KOD senaryosunun entegre edilmesinin planlandığı durumda, Yineleme 3’ün değerlendirme safhasında yapılacak kullanılabilirlik testlerine Yineleme 1 ve Yineleme 2 tamamlanmış olduğundan o yinelemelerin tüm senaryoları (A, B, C, D, E, F, G) dahil edilmektedir. Örneğin A ve F senaryolarının sistemde hiç var olmadığı ya da kullanılabilirlik testlerini geçebilecek düzeyde başarılı olmadığı sonucuna karar verilirse, bir sonraki yinelemede yer almaları öncelikli olacaktır. Böylece sistemdeki eksiklikler her yinelemede kontrol edilebilir olmaktadır. Sistem her yinelemede senaryoların yer aldığı gereksinim dokümanı sayesinde ilk günden proje tamamlanana kadar

gözlemlenebilmekte ve müşterilere sunulabilmektedir. Böylece her yinelemede kullanıcının isteklerinin eksiksiz uygulanabilmesini sağlayan, kullanıcı odaklı sistem yaratılmasına olanak sağlamaktadır.

Müşteri ve kullanıcılarla hazırlanan KOD senaryoları projedeki tüm paydaşlar arasında imzalanan bir sözleşme niteliği taşımaktadır. Hem yazılımcının hem de müşterinin sorumluluklarının korunduğu bu sözleşmeyle her iki tarafın birbirine olan güveni artmaktadır.

Tablo 4.4. Modelde yazılımcıyı koruma altına alan durum incelemeleri

Analiz ve tasarım KOD Senaryoları	Senaryo için tahmini zaman	Yorum
Yineleme 1: A senaryosunun kritik parçaları, B senaryosunun önemsiz parçaları, Yineleme 2: A Senaryosunun önemli parçaları, Yineleme 3:A Senaryosunun önemsiz parçaları entegre edilecektir.	A senaryosu 3 yineleme, B senaryosu 1 yineleme	Genellikle yazılımcı dışında kalan diğer paydaşlar kodlama basamağının uzun süren bir faz olduğundan yakınmaktadır. Ancak ihtiyaç analizlerinin çıkarılmasıyla istenen bir gereksinimin birden fazla adım sonrasında entegre edilebileceği müşteri ya da kullanıcılarca da görülmektedir. Böylece yazılımcıların hakları koruma altına alınmaktadır.

Tablo 4.4’te yer alan durumu ele alacak olursak yazılımcı dışında kalan diğer paydaşlar kodlama basamağının uzun süren bir faz olduğundan yakınmaktadır. Ancak senaryoların önceden çıkarılmasıyla, yinelemelerin kaç senaryodan oluştuğu bilgisi kullanıcı ve müşterilere verilmesiyle bu durum yazılımcıların lehine çevrilmektedir.

4.2. KOD Yazılım Geliştirme ve Fazlar

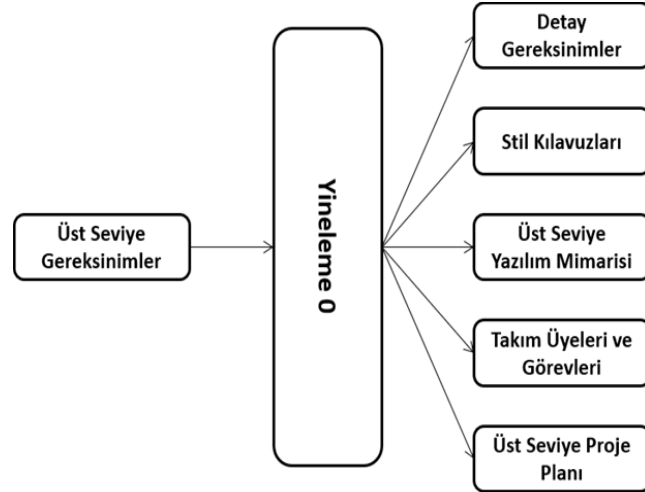
KOD yazılım geliştirme modelinde gereksinimler tamamlanıncaya kadar yinelemeler (Yineleme 0, Yineleme1, 2, 3...n şeklinde) uygulanmaktadır Her yinelemede sırasıyla analiz ve tasarım, gerçekleştirme, değerlendirme adımları uygulanmaktadır.

Yineleme 0 ise tüm aşamalardan önce sadece bir kere uygulanmaktadır. Her yineleme periyodunun süresi planlanırken çevik yazılım geliştirme yöntemlerinden UYG'nin zaman kısıtlatmalı yaklaşımı baz alınmıştır. Çevik Geliştirme Yöntemlerinden Scrum metodolojisinde yinelemeler 'sprint' adı verilen 30 günlük zaman dilimleriyle ifade edilirken UYG'de de ise bu süre projenin toplam süresi ve karmaşıklığı göz önünde bulundurularak farklılık gösterebilmektedir. Bu açıdan KOD Yazılım Geliştirme Modeli'nde de UYG (ASD) modeli referans alınarak 2-3 haftalık zaman dilimleri her bir döngünün tamamlanma süresi olarak yazılım geliştirme projesi özelinde belirlenebilecektir. Tüm yinelemeler tamamlandığında canlı ortama çıkmaya hazır olan ürün dağıtım adımıyla müşteriye sunulmaktadır.

Bu bölümde KOD yazılım geliştirme modelinin fazları her faz için tanımlanmış girdiler, fazların amaçları ve faz sonu oluşacak çıktılar ile detaylı olarak açıklanmaktadır. Ayrıca her fazın uygulanışı bir örnek ile de ifade edilmeye çalışılmaktadır.

4.2.1. Yineleme 0

En başta sistemin genel bir analizinin yapılması için başlangıç yinelemesi yani Yineleme 0 uygulanır. Yineleme 0'da, müşteriler ve analistler tarafından oluşturulan yineleme plan formu (YPF) kullanılarak gereksinim analizi yapılmaktadır. Bu formda gereksinimler ve bu gereksinimlerin sistem için öncelikleri müşteriler ve yazılım ekibi tarafından doldurulmaktadır. Daha sonra bu bilgiler, kullandıkları sistemi, şirketlerinin ve çalışanlarının yapısını en iyi değerlendirebilen analistler tarafından yorumlanmaktadır. Hangi gereksinim için ne kadar zaman biçilmeli, kimler hangi gereksinimler üzerine çalışmalı, gereksinim tipleri ve gereksinimlerin hangi yinelemede (yineleme numarası) uygulanması gerektiği bilgileri analistler tarafından doldurulmaktadır. Gereksinimler bu aşamada yazılım ekiplerine gösterilmekte ve bu konudaki değerlendirmeleri dikkate alınmaktadır. Bu aşamada yazılımcılar tarafından, gereksinimlerin sisteme entegre edileceği gerçekleştirme aşaması için yazılım mimarisi de yavaş yavaş planlanmaya başlamaktadır. Yineleme 0'ın girdi ve çıktıları Şekil 4.2' de gösterilmektedir.



Şekil 4.2. Yineleme 0 aktiviteleri

4.2.1.1. Yineleme 0 fazının girdi, çıktı ve amaçları

Girdi: Müşteri tarafından verilen üst seviye gereksinimler

Çıktı: Gruplanmış gereksinimler, Stil kılavuzları, tüm yinelemelerin planlarının doldurulduğu YPF

Amaç: Müşterilerden gelen gereksinimler öncelikle fonksiyonel/ fonksiyonel olmayan olarak sınıflandırılır. Kullanıcı Deneyimi Tasarım konusunda deneyimli kişiler tarafından stil kılavuzlarının çıkarılmasını sağlamak Yineleme 0'ın en önemli amaçlarından biridir. Böylece her yinelemede geliştirilecek prototiplerin belirli kurallarla oluşturulup, tutarlı olması sağlanmaktadır.

Örnek: Müşteriden gelen gereksinimlerin sınıflandırılıp, detaylandırılıp ve tasarım kurallarının oluşturulduğu YPF'nin bir kısmının örneği Tablo 4.5'de gösterilmektedir.

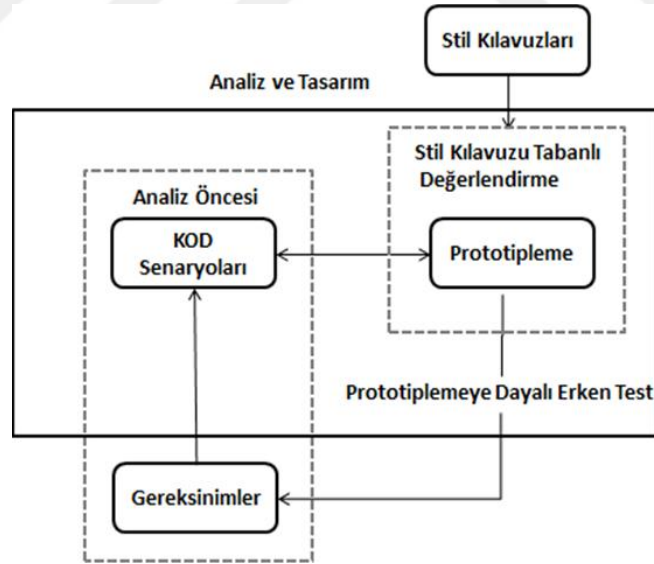
Tablo 4.5. Yineleme plan formunun bir kısmı

Gereksinim	Gereksinim Tipi	Stil kılavuzu
Ürün arama	Fonksiyonel	Benzer fonksiyonlar birbirine yakın gruplanmalı ve sık kullanılan fonksiyonlar ekranın en üstünde yer almalıdır. Ürün arama en üstte iken Benzer işlevdeki 2, 3, 4 numaralı görevler yan yana bulunmalıdır.
Ürün içerik görüntüleme		
Ürün yan etkileri gösterme		
Ürün son kullanma tarihi bilgilendirme	Fonksiyonel olmayan	Logo her ekranda sol üst köşede, CxD boyutlarında ve AxB koordinatında yer almalıdır.
Uygulamanın farklı ekranlarında uygulama logosu yer almalıdır.		

4.2.2. Analiz ve Tasarım

Yazılım projelerinin genellikle ilk ve en önemli fazlarından biri olan analiz aşaması KOD Yazılım Geliştirme Modeli'nde sistemin tasarımının bir kısmının da yapıldığı tümleşik bir yapıdan oluşmaktadır. Yapılan tasarımların değerlendirmesi ve testi de bu fazda gerçekleşen aktivitelerdendir.

Bu fazda Yineleme 0'dan gelen üst seviye gereksinimler KOD senaryolarına dönüştürülmekte ve ekran prototipleri geliştirilmektedir. KOD senaryolarına dönüştürme işleminde takım içindeki analistler ve müşteriler beraber çalışmaktadırlar. Senaryoların dönüştürülmesine paralel olarak müşteriler ya da kullanıcılarla birlikte prototipler de tasarlanmaktadır. Hazırlanan prototipler yineleme 0 da oluşturulan stil kılavuzlarına göre değerlendirilmektedir. Böylece analiz ve tasarım fazı ön analiz, prototiplemeye dayalı erken test alt fazlarından oluşmaktadır. Prototiplemeye dayalı erken test alt fazında ekranlar stil kılavuzu tabanlı değerlendirmesinde de geçmektedir. Şekil 4.3'te analiz ve tasarım fazının girdileri ve çıktıları gösterilmektedir.



Şekil 4.3. Analiz ve tasarım

4.2.2.1. Ön analiz aşaması (Pre-analiz)

Bu fazda sistemin ihtiyaçlarının gerçeği yansıtması adına gerçek kullanıcılarla ve müşterilerle irtibata geçilerek toplanan gereksinimler KOD formatıyla senaryolara

dönüştürülmektedir. Senaryolara dönüştürme işlemleri analist ekiplerince yapılmaktadır. Kullanıcı ve müşterilerin de bu dönüştürme işlemine katkı sağlaması istenmektedir. KOD senaryoları kullanıcı rollerini de içermektedir. Senaryolar, prototipleme ve erken test adımı için temel sayılacak, sistemin davranışları daha açık adımlarla ifade edeceğinden tasarım için ayrılan süre kısalmaktadır. Böylece daha kısa sürede prototipleme yapılacaktır. Prototipleme yapılırken bazı senaryolar daha somutlaşacağından KOD senaryolarından değişiklik yapma ihtiyacı görülebilecektir. Bu yüzden bu iki adım birbirini besleyerek paralel olarak ilerlemektedir.

4.2.2.2. Prototiplemeye dayalı erken test

Kullanıcılardan ayrıntılı geribildirimlerin toplanabilmesi için bu fazda senaryolar ve yineleme 0 da oluşturulan stil kılavuzları yardımıyla ekranlar tasarlanmaktadır. Bu prototipler kullanıcılar tarafından verilen gereksinimlerin doğru olup olmadığını, gereklilik ölçütünü sınamaktadır. Prototipleme ile KOD senaryolarının oluşturulması kısmı birbiriyle etkileşimli olarak devam etmektedir. Çünkü bazı durumlarda senaryoları çıkarabilmek için, sistem gereksinimleri projenin karmaşıklık seviyesine göre, kâğıt prototipleme(düşük uygunluk) ya da çevrimiçi arayüz hazırlama yazılımları(yüksek uygunluk) kullanarak tasarlanabilmektedir. Tablo 4.6’da bu iki prototipleme yönteminin kıyaslaması yapılmaktadır. Kâğıt prototiplemenin zaman tasarrufu ve düşük maliyeti, yüksek uygunluğun da gerçekçi karşılaştırmaları yapabilmesi sebebiyle modelimizde iki uygunluk yöntemi de kullanılmaktadır.

Tablo 4.6. Yüksek uygunluk ve düşük uygunluk karşılaştırması

Tip	Avantaj	Dezavantaj
Yüksek uygunluk	Alternatif tasarımlar ile gerçekçi karşılaştırmalar, kullanılabilirlik sorunlarını ortaya çıkarma	Zaman harcama
Düşük uygunluk	Zaman tasarrufu, düşük maliyet	Kullanılabilirlik testi için sınırlı kullanım, yetersiz özellikler

Ekran tasarımları geliştirilirken yineleme 0 basamağında oluşturulan stil kılavuzları baz alınarak oluşturulması önerilmektedir. Ekran tasarımlarının stil kılavuzlarına olan uygunluğunun değerlendirildiği bu alt fazda bir anlamda tasarımlar arasındaki tutarlılığın da değerlendirilmesi yapılmasını sağlamaktadır. Stil kılavuzlarına

uymayan tasarımların tekrarlanarak kurallara uygun hale getirilmesi amaçlanmaktadır. Böylece tutarlı geliştirmelerin yapılması amaçlanmaktadır.

Oluşturulan prototipler kullanılarak, analiz aşamasında dahi kullanılabilirlik testleri uygulanabilmektedir. Prototipler üzerinde Wizard Of Oz tekniği kullanılıp KOD senaryolarıyla testi yapılarak erken aşamalarda da kullanıcı odaklı yazılım geliştirmeye adım atılmaktadır. Böylece müşteri ya da kullanıcı ürünün daha ilk safhalarında ekran prototiplerini, kullanılabilirlik test sonuçlarını görebilmektedir. Erken safhalarda müşteriye sunulan prototipler ve kullanılabilirlik test sonuçları ile takıma olan güvenin artması sağlanabilecektir.

4.2.2.3. Analiz ve tasarım fazının girdi, çıktı ve amaçları

Girdi: Sınıflandırılan gereksinimler, stil kılavuzları

Çıktı: KOD senaryoları, güncel proje dokümanı (KOD senaryoları kolay okunabilir yapısıyla doküman formatına kolayca dönüştürülebilir.)

Amaç: Toplanan gereksinimlerin müşteriler ve kullanıcılar tarafından her yinelemede KOD senaryosu formatına dönüştürülmesi, yineleme 0 da oluşturulan stil kılavuzlarına göre tasarımların yapılması ve bu aşamada oluşturulan prototiplerle kullanılabilirlik testlerinin erken safhalarda uygulanabilmesi.

Örnek: Analiz ve tasarım aşamasının örneklendirmesi Tablo 4.7’de gösterilmektedir.

Tablo 4.7. Analiz ve tasarım

Gereksinim	DOG(BDD) senaryosu	Prototip
İstediğim ürünün barkod numarasını arama kutusuna girerek içindekiler listesine ulaşmak istiyorum.	Scenario Outline: Ürün barkod numarası ile içindekiler gösterme Given: Arama kutusunun bulunduğu anasayfadayım. And: İlgili alana barkod numarası girip When:”Ara” butonuna ya da “Enter”e bastığımda Then: İçindekiler listesi sıralanmalı.	İlgili senaryoya ait kâğıt prototipleme ya da çevrimiçi yazılım araçları kullanılarak, stil kılavuzu uygun şekilde tasarımı geliştirilebilmektedir.

4.2.3. Gerçekleştirme

DOG (BDD), KOD Yazılım Geliştirme Modeli’nde sadece analiz fazında değil, gerçekleştirme fazında da KOD senaryolarının kullanılması ile uygulanmaktadır.

Yazılımı kolayca yapılandırmak, kusursuz kodlamayı etkinleştirmek için analiz sırasında kullanılan KOD senaryoları bu fazda da kullanılmaktadır. Modelin bu fazında KOD senaryoları kullanılarak sınıf testleri ve arayüz testlerinin yazılmasına odaklanılmaktadır. Ayrıca bazı yazılım araçları kullanılarak (Specflow, WatIN) otomatize testler geliştirilebilmektedir. Yazılan test metotların başarısız olması durumunda, yazılımın hatalı olduğu anlaşılmaktadır. Böylece kullanıcılara ulaştırılacak kodun her zaman ayıklanmış, hatasız olması amaçlanmaktadır. Hatalı kod düzeltilene kadar modelin test fazına geçilememektedir.

4.2.3.1. Gerçekleştirme fazının girdi, çıktı ve amaçları

Girdi: KOD senaryoları

Çıktı: Geçerli kod, otomatize kullanıcı arayüzü testleri, sınıf ve arayüz testleri

Amaç: Analiz ve tasarımdan gelen KOD senaryolarının kod yazımında kullanılmasıyla geçerlemelerinin yapılması, aynı senaryolarla otomatize edilmiş testlerin yazılması, otomatik olarak kodların çalıştırılması, kullanıcı arayüz testlerinin çalıştırılması

Örnek: Gerçekleştirme aşamasının örneklendirmesi Tablo 4.8’de gösterilmektedir.

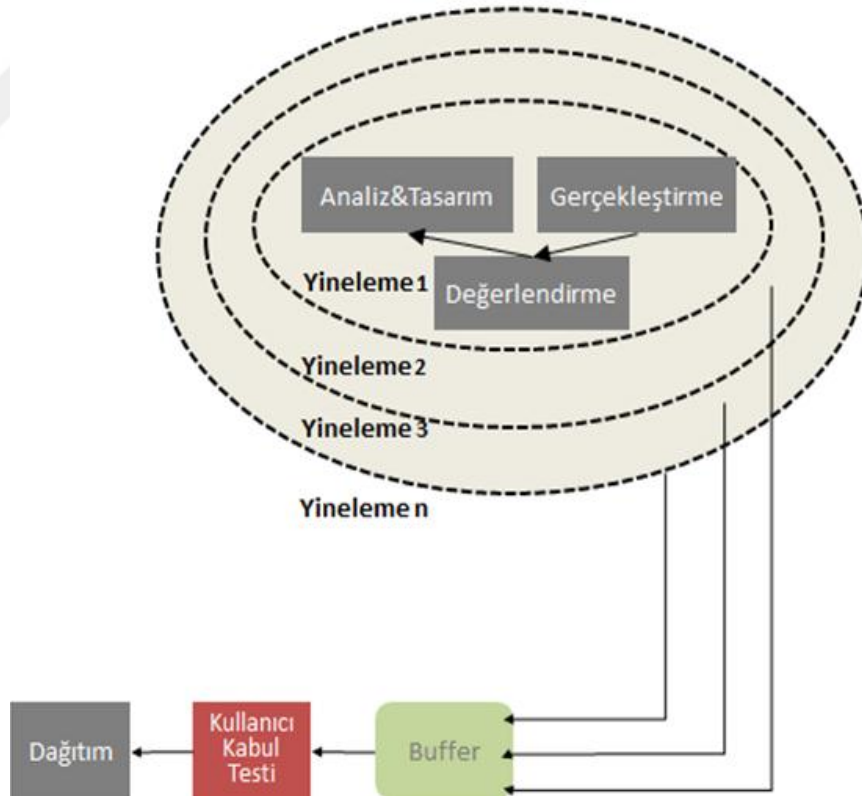
Tablo 4.8. Gerçekleştirme aşaması

Gereksinim	DOG(BDD) senaryosu	Kod
İstediğim ürünün barkod numarasını arama kutusuna girerek içindekiler listesine ulaşmak istiyorum.	Scenario Outline: Ürün barkod numarası ile içindekiler gösterme Given: Arama kutusunun bulunduğu anasayfadayım. And: İlgili alana barkod numarası girip When: "Ara" butonuna ya da "Enter"e bastığımda Then: İçindekiler listesi sıralanmalı.	Scenario Outline: Ürün barkod numarası ile içindekiler gösterme Given: Arama kutusunun bulunduğu anasayfadayım. And: İlgili alana <barkod numarası> girip When: "Ara" butonuna ya da "Enter"e bastığımda Then: İçindekiler listesi sıralanmalı. Examples: barkodnumarasi B456G99

4.2.4. Değerlendirme

Değerlendirme fazı modelin merkezinde yer almaktadır. Sistemde karar mekanizması oluşturularak bir sonraki adımın hangi faza dahil edileceğinin kararı burada verilmektedir.

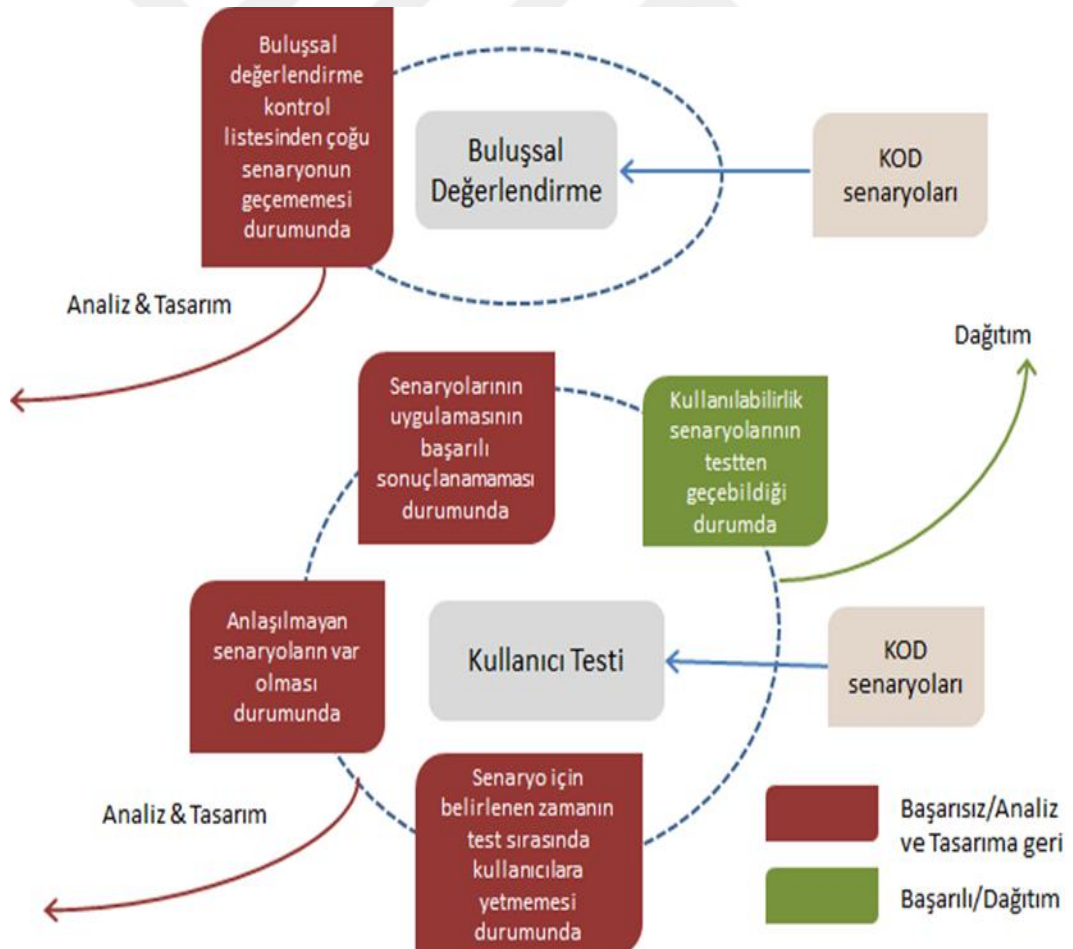
Şekil 4.4'te gösterildiği üzere her yineleme sonrasında dağıtım için karar verilen kod parçaları bufferda toplanmaktadır. Buffer her yineleme sonrasında sürümler için biriktirilen, müşterilere sunulmak istenen, kodlanan ve testlerden geçmiş başarılı senaryoların bütünüdür. Böylece artımlı olarak çoğalarak bir sonraki adım için hazır beklemektedir. Değerlendirme fazında geçerlemede başarısız olan kodlar (senaryolar) ise tekrar analizinin yapılması için analiz ve tasarım fazına geri gönderilmektedir. Bufferda biriken senaryolar Kullanıcı Kabul Testi'ne (User Acceptance Test) tabi tutularak, kullanıcılar ve müşteriden onay alınması sonrasında dağıtım için hazır olmaktadır. Şekil 4.4' te durum özetlenmektedir.



Şekil 4.4. Değerlendirme fazı

Değerlendirme fazında yapılan testlerde uygulanabilecek kullanılabilirlik değerlendirme yöntemleri stil kılavuzları, sezgisel değerlendirme, kullanıcı testleri ve

model temelli yaklaşım şeklinde gruplandırılabilir (Çağıltay, 2011). Bu yöntemlerden sezgisel değerlendirme, stil kılavuzu tabanlı değerlendirme ve kullanıcı testleri modelde kullanılmakta olup, sezgisel değerlendirme için uzmanlar, kullanıcı testleri içinse hedef kullanıcılar ile temsili kullanıcılar üzerinde uygulanarak, sonuçları değerlendirilmektedir. Test edilecek senaryolar, analiz sonrasında oluşturulmuş KOD senaryolarından oluşmaktadır. Böylece tüm gereksinimler kullanılabilirlik testine tabi tutulmaktadır. Sisteme süreç içerisinde yeni gereksinimler dahil edilmek istenebilir. Gereksinim analizine yeni dahil olan bir senaryo kodlandıktan sonra test aşamasına gelindiğinde, ilgili KOD senaryosuyla birlikte kullanılabilirlik testine tabi tutulmaktadır. Böylece çıkarılan her yeni gereksinimde kullanıcının davranışlarının değerlendirilebildiği bir süreç ortaya çıkmaktadır. Dağıtım fazına gelmeden birçok kez kullanılabilirlik testi yapılan yazılım daha tutarlı bir hale dönüşmektedir.



Şekil 4.5. Değerlendirme kriterlerinden bazıları

Şekil 4.5'te modelin değerlendirme fazında uygulanan karar mekanizması gösterilmektedir. Değerlendirme fazında yapılan kullanılabilirlik test sonuçları yorumlanırken Tablo 4.9'da gösterilen yaklaşım uygulanmaktadır. Karar tablosu uygulanan kullanılabilirlik yöntemine göre değişim göstermektedir. Çalışanlar tarafından belirlenen öncelikler karar mekanizmasını şekillendirmektedir.

Tablo 4.9. Karar tablosu

Kullanılabilirlik testi	Karar için bulgular	Karar	Sonuç
Sezgisel (Heuristic) Değerlendirme	Sezgisel kontrol listesinden senaryoların geçmesi	Dağıtım hazır	Dağıtım
Kullanıcı testi	Kullanılabilirlik senaryolarının testten geçebildiği durumda	Dağıtım hazır	Dağıtım
	Anlaşılmayan senaryolar var olması durumunda	Analize geri gönderme	Analiz
	Senaryo için belirlenen zamanın test sırasında kullanıcılara yetmemesi durumunda	Analize geri gönderme	Analiz

Kullanılabilirlik testleri sonrasında bir problemle karşılaşıyorsa tekrar analizinin yapılması için analiz aşamasına geri dönülmektedir. Eğer senaryolar belirlenen tekniğe göre başarılı bir şekilde uygulandıysa bir sonraki adım dağıtım adıdır.

4.2.4.1. Değerlendirme fazının girdi, çıktı ve amaçları

Girdi: KOD senaryoları, senaryoların entegre edildiği yazılım

Çıktı: Kullanılabilirlik test sonuçları

Amaç: İlk fazda müşteriler ve kullanıcılarca hazırlanan KOD senaryolarının kullanılabilirlik testinde kullanılarak fazın kullanıcı odaklılığı açısından değerlendirilmesinin yapılabilmesi. Geliştirilen senaryoların kullanılabilirlik açısından başarısının değerlendirilmesiyle bir sonraki adımın belirlenmesi.

Örnek: Değerlendirme aşamasının örneklendirmesi Tablo 4.10'da gösterilmektedir.

Tablo 4.10. Değerlendirme aşaması

KOD senaryosu	Kullanılabilirlik senaryosu
Scenario Outline: Ürün barkod numarası ile içindekiler gösterme Given: Arama kutusunun bulunduğu anasayfadayım. And: İlgili alana barkod numarası girip When: "Ara" butonuna ya da "Enter"e bastığımda Then: İçindekiler listesi sıralanmalı.	"Sayfada ilgili arama alanına ürün barkod numarası girildiğinde, ürünün içerik bilgisinin görüntülenebilmesini sağla" görevini kısa sürede ve kullanıcıyı tatmin edici şekilde gerçekleştirmenin kontrolü

4.2.5. Dağıtım (Deployment)

Dağıtım, değerlendirme basamağı sonrasında üretim ortamına koyulmak üzere bufferda tutulan ve müşteriye sunulması istenen kodların (senaryoları) paketlenme işlemidir. Bir yönüyle bu aşama müşterilere sunulacak ürünün yaratıldığı son fazdır. Bu aşama sonrasında tutarlı sürümlerin kullanıcılara ulaştırılması en öncelikli amaçlardandır. Bu fazdan önce Kullanıcı Kabul Testi uygulanarak son ürün üzerinde müşterilerinin onayının alınması sağlanmaktadır.

Kullanıcı Kabul Testi kısaca son kullanıcı testi olarak bilinmektedir (User Acceptance Testing (URL-9, 2015) Nihai ürünün müşteriye sunulmadan önce son kez, son kullanıcılar tarafından test edilmesi ürünün canlı ortamda karşılaşacağı sorunları önceden görüp engellemek adına yapılmaktadır. Kullanıcı Kabul Testleri'nin Alfa ve beta kabul testleri gibi türleri bulunmaktadır. Alfa testi testi yapacak kullanıcıların geliştirme yapılan yerde sistemi test etmeleridir. Genellikle ürünü organizasyonda çalışan kişiler test etmektedir. Ayrıca potansiyel kullanıcı grubuyla beraber de yapılabilmektedir ancak önemli olan nokta testin, geliştirmenin yapıldığı yerde gerçekleştirilmesidir. Alfa testi kullanıcılarından toplanan geribildirimlerle belirli sorunları ve hataları düzeltilip, ürünün kullanılabilirliğini arttırmak amaçlanmaktadır (URL-9, 2015). Alan testi olarak da bilinen Beta Testi, müşteri ortamında gerçekleştirilmektedir. Beta kullanıcıları ürünün gelişimine katkı sağlamak için geribildirimlerde bulunmaktadır. İki testte de özetle amaç müşteriye çıkmaya hazırlanan ürünün son kez teste tabi tutularak olası problemleri önceden görüp ürünü düzeltmektir.

4.2.5.1. Dağıtım fazının girdi, çıktı ve amaçları

Girdi: Her yinelemenin değerlendirme adımı sonrasında müşteriye çıkmaya hazır olan başarılı kodların toplu bir şekilde Kullanıcı Kabul Testi'nden geçmiş, nihai versiyonu.

Çıktı: Müşteriye çıkabilecek kod

Amaç: Müşterilerin tüm gereksinimlerinin yer aldığı, kullanıma hazır sistemin oluşturulması.

4.3. KOD Yazılım Geliştirme ve Yinelemeleri

KOD Yazılım Geliştirme Modeli'nin yinelemeleri UYG (ASD) modelinin zaman kısıtlı yaklaşımı baz alınarak oluşturulmuştur. Her yinelemede geliştirme adımlarına harcanan zaman farklılaşmaktadır. Çünkü bir projede tüm adımlar sıralı bir şekilde işletilmemektedir. Örneğin bir döngüde analiz yapılırken bir yandan da önceki döngülerden kalmış bir geliştirme (kodlama) yapılıyor olabilmektedir. UYG (ASD) modeli referans alınarak 2-3 haftalık zaman dilimleri her bir yinelemenin tamamlanma süresi olarak belirlenmiştir. Birbiriyle paralel ilerleyen geliştirme adımları Şekil 4.6'da gösterilmektedir.

Geliştirme Adımı		Yineleme0	Yineleme1	Yineleme2	Yinelemen
Stil kılavuzu oluşturma					
Analiz ve Tasarım	Ön analiz				
	Prototiplemeye dayalı erken test				
Gerçekleştirme					
Değerlendirme					
Dağıtım					

Şekil 4.6. Döngülerde geliştirme adımlarına harcanan zaman

Yineleme 0 adımı modelde ilk fazı olarak düşünülmüştür ve sadece bir kez uygulanır. Bu fazda modelin tüm gereksinimleri genel olarak analiz edilip, stil

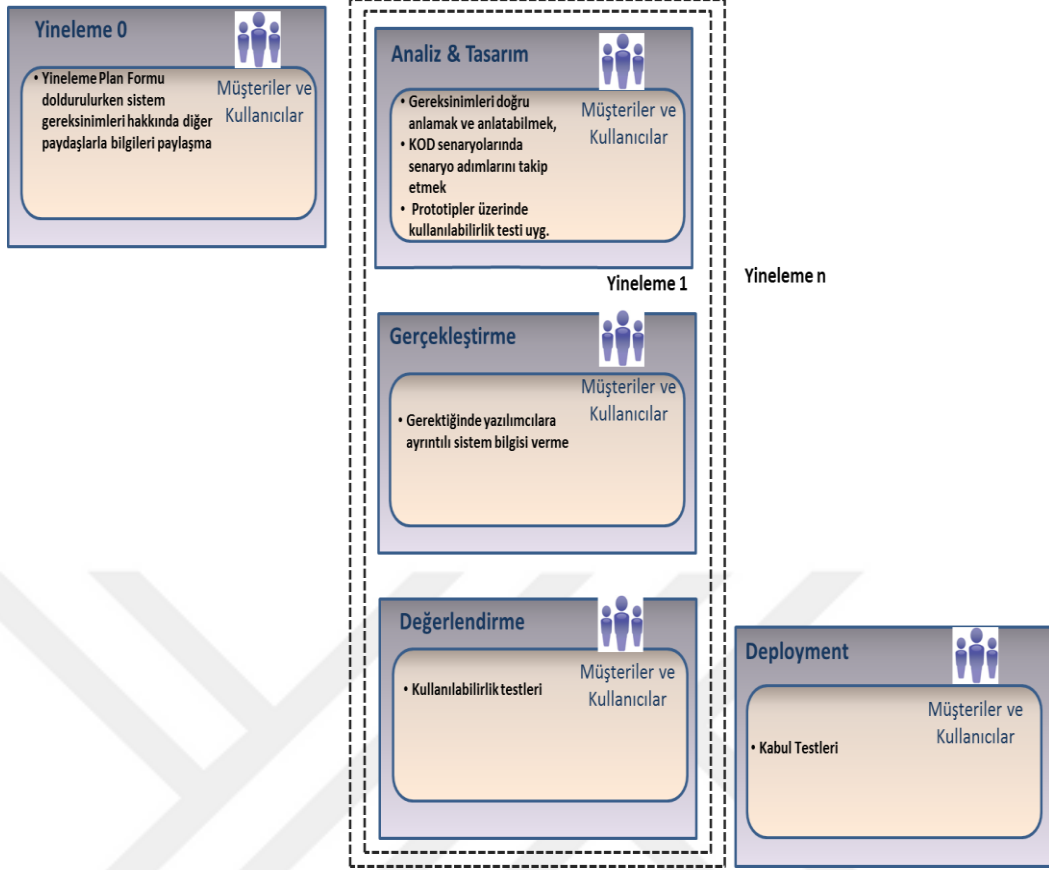
kılavuzları oluşturulduğundan yineleme 0 da analiz ve stil kılavuzu oluşturma dışında diğer geliştirme adımları uygulanmamaktadır. Analiz ve tasarım adımı ilk yinelemelerde daha yoğun, gerçekleştirme ve değerlendirme daha seyrek uygulanırken, son yinelemelere doğru analiz ve tasarım yoğunluğu azalır, değerlendirme ve gerçekleştirmenin yoğunluğu artmaktadır. Dağıtımın gerçekleştiği yinelemeden sonraki yinelemede gereksinim analizin yapılabilmesi için Yineleme 0 tekrar uygulanabilmektedir.

4.4. KOD Yazılım Geliştirme ve Roller ve Sorumluluklar

4.4.1. Müşteriler ve kullanıcılar

Müşteriler ve kullanıcılar Şekil 4.7’de gösterildiği üzere önerilen KOD Yazılım Geliştirme Modeli’nin neredeyse her safhasında yer almaktadır. Modelde kullanılan DOG(BDD) yaklaşımı müşteriye ve kullanıcıyı anlamayı kolaylaştırmayı ve her zaman bu iki grubun isteklerini önde tutmayı amaçlamaktadır. Müşteri ve kullanıcıların modeldeki rolünü özetlersek:

- Gereksinimleri doğru anlamak ve anlatabilmek,
- Modelde yer alan diğer paydaşlara (analist, test ve yazılım ekibine) gerekli bilgiyi zamanında verebilmek, onlarla beraber çalışabilmek,
- Yineleme 0 fazında yineleme plan formunu doldururken sistem gereksinimleri hakkında gerekli bilgileri vererek paydaşlara yardımcı olmak,
- Analiz ve tasarım fazında gereksinimlerin KOD senaryolarına dönüştürülmesi sırasında senaryo adımlarını takip edip, kontrollerini sağlamak
- Analiz ve tasarım fazında oluşturulan prototipler üzerinde kullanılabilirlik testlerini uygulamak ve geribildirimlerde bulunmak,
- Değerlendirme fazında kullanılabilirlik testlerine katılıp, geribildirimlerde bulunmak



Şekil 4.7. Müşteri ve kullanıcı rolleri

4.4.2. Analist ekibi

KOD Yazılım Geliştirme Modeli'nde müşteri ve kullanıcılar kadar analistlerin de rolü oldukça büyüktür. KOD son sürümün ilk aşaması olan Yineleme 0'da gereksinimler kullanıcı ve müşteriler yardımıyla belirlenmektedir. Belirlenen gereksinimler fonksiyonel/fonksiyonel olmayan şeklinde sınıflandırılırken analistlerin de görüşü alınmaktadır. Böylece modelin değerlendirme fazında yapılacak kullanılabilirlik testlerinde fonksiyonel olmayan gereksinimlerin önceliklendirilmesi ve daha çok test edilmesi sağlanacaktır.

Müşterilerce belirlenen gereksinimlere ilişkin detayların yer aldığı Yineleme Plan Formu'nun (YPF) doldurulması, sistemi daha iyi tanıyan analistler tarafından yönetilmekte ancak müşteriler de bu süreçte yer almaktadır. Gereksinimlerin öncelik sırası ve yineleme periyotlarının belirlendiği, müşteri ve analistlerin birlikte çalışarak oluşturdukları YPF'nin bir örneği Tablo 4.11'de gösterilmektedir.

Tablo 4.11. Yineleme X için örnek yineleme plan formu

Örnek Gereksinim	Öncelik	Tip	Max (gün)	Yineleme	Tip/Grup
Kullanıcı sistemden çıkış butonuna 2 kez bastıktan sonra çıkmasına izin vermesi	Düşük	Hata Düzeltme	20	Yin.1	Fonksiyonel
Kullanıcılara yanlış besin grubunun gösterilmesi	Yüksek	Hata Düzeltme	10	Yin.1	Fonksiyonel
Kullanıcının listelediği besinleri mail atabilmesi	Düşük	Yeni Geliştirme	2	Yin.2	Fonksiyonel
Uygulama logosu her ekranda sol üst köşede, AxB koordinatında yer almalıdır.	Düşük	Geliştirme	1	Yin.2	Fonksiyonel olmayan

Yineleme 0’da belirlenen, gruplandırılan ve detaylandırılan gereksinimler, Analiz ve Tasarım aşamasında KOD senaryolarına dönüştürülmektedir. KOD senaryosu yazımı teknik anlamda çok tecrübe gerektirmese de sadece müşteri ya da kullanıcılar tarafından yazılması kolay olmayacağından analistlerin bu görevi kullanıcılar/müşteriler ile beraber üstlenmesi daha doğru bulunmuştur.

4.4.3. Test ekibi

Organizasyonlarda yer alan test ekipleri içerisinde kullanıcılar, müşteriler ya da kendi bünyesinde hizmet veren test mühendisleri yer almaktadır. Test ekibinin görevi organizasyonda geliştirilen yazılım projelerinin müşteriye sunulmadan önce testini yapıp, en az hata ile müşteri ve kullanıcılara sunulmasını sağlamaktır.

KOD v.2’ nin Analiz ve Tasarım aşamasında oluşturulan kâğıt ya da çevrimiçi prototipler ile sistemin ilk kullanıcı testleri yapılabilmektedir. Müşteriler ve kullanıcılardan gelen gereksinimler KOD senaryolarına dönüştürüldükten sonra, kullanılabilirlik testi senaryosu olarak da kullanılabilmektedir. Ayrıca model, daha ilk aşamalarda dahi, prototipler üzerinden Wizard Of Oz tekniği ile test ekibi tarafından test edilebilmektedir. KOD son sürümde her yinelemenin değerlendirme fazında sistemin kullanılabilirlik testleri yapılmaktadır. Bu testlerde test ekibi ile birlikte gerçek kullanıcılar ve müşterilerin de yer alması önerilmektedir. KOD v.2’nin son aşaması olan dağıtım fazından önce son kullanıcı kabul testleri

yapılmaktadır. Test ekibi bu son test ile ürünün müşteriye hatasız bir şekilde ulaşmasını amaçlanmaktadır.

4.4.4. UX uzmanı

KOD v.2'nin Yineleme 0 aşamasında UX Uzmanı ya da UX konusunda deneyimli kişilerle çalışılmaktadır. Çünkü bu aşamada müşteri ve kullanıcılardan gelen gereksinimler değerlendirildikten sonra UX uzmanı kişiler tarafından stil kılavuzları çıkartılmaktadır. Stil kılavuzları prototiplerin standardize edilmesine yardımcı olmaktadır. Böylece bu yaklaşım ekranlar arası tasarımlarda tutarlılığı sağlamaktadır (Mayhew, 1999).

Türkiye'deki organizasyonların kullanılabilirlik ve yaşam döngüsü durumunu belirlemek amacıyla yapılan ankette ve yüz yüze yapılan görüşmelerde kişilerin çalıştıkları organizasyonlarda kullanılabilirlikle ilgili unvanı olan kişiler hakkında bilgi alınmıştır. Organizasyonlarda genelde bu sorumluluğu paylaşan kişilerin arayüz tasarımcıları, kullanıcı deneyimi tasarımcıları ve ürün tasarımcıları oldukları görülmüştür. Yapılan alanyazın çalışmaları sonrasında Dünya'da yer alan organizasyonların durumu bu konuda ülkemize göre daha iyi gözükse de yine de beklenen aşamaya gelinmediği görülmektedir (Gulliksen, 2004). Önerilen modelde bu soruna çözüm olarak daha az zaman ve kaynak gerektirdiğinden İndirimli Kullanılabilirlik Mühendisliği (Discount Usability Engineering) önerilmiştir (Schumacher, 2010). Bu yöntemde 3 teknik uygulanabilmektedir:

Senaryolar: Sistemin gereksinimlerini ifade eden basit ve kısa senaryolar.

Basitleştirilmiş sesli düşünme: Senaryoları gerçekleştirirken katılımcıların sesli yorumlar yapmasını sağlamak, düşüncelerini öğrenmek.

Sezgisel değerlendirme Bu metotta aşağıda gösterilen 10 temel sezgisellerin kullanılması önerilmiştir (Nielsen, 1994).

- Görünürlük
- Eşleşme
- Kullanıcı kontrolü
- Tutarlılık

- Hata önleme
- Hatırlama
- Esneklik ve verimlilik
- Estetik ve minimalist tasarımı
- Hata kurtarma
- Yardım ve dokümantasyon

4.4.5. Yazılımcılar

KOD v.2'nin Gerçekleştirme fazında yazılımcılar görev almaktadır. Yazılımcılar kodlama sırasında, Yineleme 0'da gelen gereksinimlerden oluşturulan senaryoları baz almaktadır. Yineleme 0'da YPF'nin müşteri, kullanıcı ve analistler tarafından doldurulması sırasında yazılımcıların senaryolar için harcayacağı zamanlar hakkında görüşleri alınmaktadır. Bununla birlikte diğer fazlarda da sistemle ilgili yazılım mimarisi hakkında bilgi sahibi olunmak istendiğinde yazılım grubundakilerden yardım talep edilebilmektedir.

5. SONUÇLAR

Bu çalışmada İBE ve YM alanlarını birleştirerek kullanıcı odaklı sistemlerin oluşturulabilmesine olanak sağlamak, müşteriler, kullanıcılar ve yazılımcıların ortak dilden konuşup anlaşabilmesi ve böylece sistemin gereksinimlerinin iyi analiz edilip minimum yanlışla geliştirilebilmesi amaçlanmaktadır.

5.1. Yapılan Çalışmanın Özeti

Günümüzde kullanılabilirliğin yazılım projelerindeki önemi gittikçe artmaktadır. Bu yüzden İBE ve YM alanlarının birbirine entegre edildiği birleşik sistemler günümüzde sıkça önerilmektedir. Yazılım mühendisliği sistemin daha çok fonksiyonel gereksinimlerini gidermeye çalışırken, İBE kullanıcıya odaklanmaya çalışmaktadır. Ancak incelenen alanyazın çalışmaları da iki alanın birbiriyle nasıl birleşeceğini ve nasıl aynı dilde konuşup entegrasyonunun gerçekleşeceğini gösteren çalışma sayısı azdır. Ayrıca alanyazındaki çalışmalara göre önerilen modelde gereksinim analizi için yapılan çalışmalar DOG sayesinde sağlıklı bir şekilde yürütülmektedir. Sistemde gerçekten yer alması gereken gereksinimlerin müşteri ve kullanıcılar ile belirlenmesi sağlanmakta DOG (BDD) ile analist ekibi ile birlikte bu gereksinimlerin senaryolara dönüştürülmesi sağlanmaktadır.

Kullanılabilirliğin yazılım projelerindeki önemi arttığından önerilen modelin kullanıcıya yakınlığı her aşamada gözden geliştirilmiştir. Müşteri ve kullanıcının isteklerine odaklanan yaklaşımlar araştırılmıştır. DOG (BDD) kullanılarak projeye dahil olan tüm paydaşların birbiriyle olan iletişimi sağlıklı bir şekilde gerçekleştirilmektedir(URL-1, 2014). Ayrıca projedeki her gereksinim senaryolarla ifade edildiğinden bir anlamda projenin dokümantasyonu da yaratılmış olmaktadır. Önerilen modelde her yinelemede işletilecek senaryoların belli olmasından ve bu bilgilerin güncelliğinin korunmasından dolayı oluşturulan bu dokümantasyonun da güncel olması sağlanmaktadır. Önerilen modelin ilk aşaması olan Yineleme 0'da kullanıcı ve müşterilerden toplanan üst seviye gereksinimler incelenmektedir. Fonksiyonel olan/olmayan gereksinimler gruplandırılarak detaylandırılması

sağlanmaktadır. Fonksiyonel olmayanlar gereksinimler UX uzmanları tarafından gözden geçirilmektedir. Ekranlar arasında tutarlılığın sağlanması için stil kılavuzları çıkarılmaktadır. Analiz ve Tasarım fazında yapılacak tasarımlar bu stil kılavuzlarına göre oluşturulmaktadır.

Önerilen modelin Analiz ve Tasarım aşamasında prototipler yineleme 0 da oluşturulan stil kılavuzlarına göre oluşturularak kullanılabilirlik testleri yapılabilmektedir. Stil kılavuzlarına göre oluşturulmayan prototiplerin tekrarlanması önerilmektedir. Prototiplerin tamamlanmasından sonra ayrıca Wizard Of Oz tekniği kullanılarak senaryoların canlandırılması sağlanabilmektedir. Böylece yazılım sürecinin erken safhalarında kullanıcı odaklı sistem oluşturmak için çalışmalar yapılmış olmaktadır.

Gerçekleştirme fazında KOD senaryolarını kullanarak test metotları yazılmaktadır. Böylece kodların hatasız geliştirilmesi amaçlanmaktadır. Değerlendirme fazında da sistemin kullanıcı testleri ve sezgisel değerlendirmesi yapılarak kullanıcı odaklı sistemin yaratılması amaçlanmaktadır. Ayrıca değerlendirmeden dağıtım fazına geçişte de son kez kullanıcı kabul testleri yapılarak sistemin son kez değerlendirilmesi ve eksiklerinin giderilmesi amaçlanmaktadır.

5.2. KOD Yazılım Geliştirme Modeli Özeti

KOD Yazılım Geliştirme Modeli İBE ile YM alanlarını birleştirmek için önerilen bir modeldir. Modelin karakteristik özellikleri aşağıda sıralanmıştır:

- KOD Yazılım Geliştirme Modeli yinelemeli yaklaşımla oluşturulmuştur.
- Başlangıçta ve tek sefer uygulanan Yineleme 0 uygulanmaktadır. Daha sonra yinelemeli olarak sırasıyla analiz ve tasarım, gerçekleştirme, değerlendirme ve dağıtım fazları uygulanmaktadır.
- Modelin yinelemeleri UYG modeli baz alınarak zaman kısıtlı olarak tanımlanabilmektedir. Böylece her yinelemede geliştirme adımları paralel olarak uygulanmaktadır.
- Yazılım yaşam döngüsü boyunca müşteriler ve kullanıcılarla sürekli iletişim halinde olup gereksinim analizi sırasında onlarla çalışılmaktadır.

- DOG (BDD) çevik yaklaşımı kullanılarak yaşam döngüsünde yer alan tüm paydaşların ortak dilde konuşması sağlanmaktadır. Teknik tarafta çalışan kişilerle yönetim(business) tarafındaki kişilerin arasındaki iletişim problemleri minimuma indirilmektedir.
- DOG (BDD) kullanılarak otomatize test metotları yazılabilmektedir.
- Stil kılavuzları modelde kullanılıp tasarımlar bu kılavuza göre oluşturulmaktadır. Böylece ekranlar arasında tutarlılık sağlanmaktadır.
- Toplanan gereksinimler modelin yineleme 0 başlangıç fazında UX uzmanları tarafından fonksiyonel/fonksiyonel olmayan olarak gruplandırılarak stil kılavuzları oluşturulmaktadır. Ancak organizasyonlarda UX uzmanları bulunmuyorsa, buna çözüm olarak İndirimli Kullanılabilirlik Mühendisliği önerilmektedir.
- Modelin daha ilk aşamalarında oluşturulan prototipler üzerinden kullanıcı testleri yapılabilmekte böylece kullanılabilirlik diğer modellerdeki gibi sadece son fazlarda uygulanan bir aktivite olarak görülmemektedir.

5.3. KOD Yazılım Geliştirmenin Katkısı

KOD Yazılım Geliştirme Modeli'nde İBE ve YM alanındaki iyi pratikler entegre edilmeye çalışılmıştır. DOG (BDD) yaklaşımının kullanımıyla KOD senaryolarının yazımında iş birimlerinin de dahil olması sebebiyle senaryolar yazılımcılar, analistler ve müşteriler arasındaki iletişimi kolaylaştırmakta ve teknik detaylardan daha çok sistem davranışı hakkında bilgi vermektedir. Böylece tüm paydaşların iletişimi sağlıklı bir şekilde yürütülmektedir. KOD Yazılım Geliştirme Modeli test metotlarının kolayca entegre edilebildiği esnek bir model örneğidir. Bazı yazılım araçlarının kullanımıyla KOD senaryoları üzerinden kolayca otomatize testler de yazılabilmektedir. Ayrıca KOD senaryolarıyla müşterilerin ve kullanıcıların anlayacağı dille çalıştırılabilir test metotları da yazılmış olacaktır.

KOD Yazılım Geliştirme Modeli birçok kullanılabilirlik tekniğinin kullanımını içinde barındıran, kullanıcıyı her zaman merkezde tutmaya çalışan bir modeldir. Stil kılavuzları, kullanıcı testleri, Wizard Of Oz, sezgisel değerlendirme gibi aktivitelerle modelin son aşamalarında değil ilk aşamalarında da uygulanarak geleneksel yazılım yaşam döngülerinden farkını ortaya koymaktadır.

KOD senaryolarının yazımı gereksinim analizlerinin detaylı notları niteliğindedir. Böylece her yinelemede senaryoların varlığıyla dokümantasyon kendiliğinden sağlanmış olmaktadır. KOD Yazılım Geliştirme Modeli'nin sağladığı tüm katkılar, projelerde kolay uygulanabilirliğiyle birlikte piyasada çalışan kişilere öneri olarak sunulabilecektir.

5.4. Çalışmanın Kısıtları

Önerilen modelin katkılarıyla birlikte kısıtları da bulunmaktadır. Modelin uygulanabilirliğinin değerlendirilmesi çalışması yüksek lisans öğrencileri tarafından gerçekleştirilmiştir. Ancak katılımcıların tecrübe farklılıklarından dolayı değerlendirme çalışmasında modelin tüm fazlarının uygulanması yapılamamıştır. Sadece analiz ve tasarım kısmını gerçekleştiren katılımcıların bu konudaki görüşleri değerlendirilmiştir.

Çalışmanın bir diğer kısıtı ise yapılan anket ve görüşmelerin az sayıda katılımcı ile gerçekleştirilmesidir. Daha fazla sayıda katılımcının varlığı daha genelleştirilebilir sonuçlara götüreceğinden katılım konusunda çalışmanın kısıtının olduğu söylenebilmektedir.

İhtiyaç analizi sırasında yapılan anket ve görüşmelerde katılımcıların geliştirdikleri yazılım türlerine göre cevapları incelenmemiştir. Yazılımların türleri önerilen modelin geliştirilmesi sürecinde katkı sağlayacağından, katılımcılardan bu konuda bilgi alınmaması bir eksikliktir.

Modelin klasik modellerle karşılaştırılmasının yapıldığı başka bir değerlendirme çalışması mevcut değildir. İBE ve YM alanlarının entegre edildiği diğer tümleşik sistemler ile KOD Yazılım Geliştirme Modeli'nin fazlarının farkını gösterecek, avantaj ve dezavantajlarının sıralanabildiği bir karşılaştırma çalışması mevcut değildir.

5.5. Gelecek Çalışmalar

Önerilen model üzerinde gelecekte uygulanabilecek aktiviteler modelin kısıtlarıyla bağlantılıdır. KOD Yazılım Geliştirme Modeli'nin uygulanması tüm fazlar üzerinde

gerçekleştirilememiştir. Gelecekteki çalışmalara tüm fazların uygulanmasının gerçekleştirilebileceği bir değerlendirme uygulaması dahil edilmelidir.

KOD Yazılım Geliştirme Modeli'nin kurumlar içerisinde uygulanmakta olan yazılım geliştirme yöntemleri ile karşılaştırılma çalışması eksik olduğundan, bir karşılaştırma çalışması yapılmalıdır Genelde uygulanan yazılım geliştirme modelleri ile önerilen modelin performans kıyaslamasının yapılabildiği, dokümante edilebildiği bir yazılım geliştirme projesinden sonuçlar elde edilmelidir.



KAYNAKLAR

Anderson J., Fleek F., Garrity K., Drake F., Integrating Usability Techniques into Software Development, *IEEE Software*, 2001, **18**(1), 46– 53.

Ardito C., Buono P., Caivano D., Costabile M., Lanzilotti R., Bruun A., & Stage J., *Usability Evaluation: A Survey of Software Development Organizations*, SEKE, Miami, FL, USA, 2011.

Astels D., *Test-Driven Development: A Practical Guide*, 1st ed., NJ:Prentice Hall , Upper Saddle River, 2003.

Bak J. O., Nguyen K., Risgaard P., Stage J., *Obstacles to Usability Evaluation in Practice: A Survey of Software Development Organizations*, 5th Nordic Conference on Human-Computer Interaction, Lund, Sweden, 2007.

Beck K., *Test Driven Development: By Example*, 1st ed., Newyork, Addison-Wesley, 2003.

Birant K. U., Boz E., *Yazılımda Kullanılabilirlik Analizlerinin Türkiye’de Kullanımı*, 5. Ulusal Yazılım Mühendisliği Sempozyumu, Ankara, 26-28 Eylül 2011.

Bygstad B., Ghinea G., Brevik E., Software Development Methods and Usability: Perspectives From a Survey in the Software Industry in Norway, *Interacting With Computers*, 2008, **20**(3), 375-385.

Chamberlain S., Sharp H., Maiden N. 2006, Towards a Framework for Integrating Agile Development and User-Centered Design, *Extreme Programming and Agile Processes in Software Engineering*, DOI:10.1007/11774129_15.

Chilana P. K., Ko A. J., Wobbrock J. O., Grossman T., Fitzmaurice G., Post-deployment Usability: a Survey of Current Practices, *SIGCHI Conference on Human Factors in Computing Systems*, Florence, Italy, 5-10 April 2008.

Çağıltay K., *İnsan Bilgisayar Etkileşimi ve Kullanılabilirlik Mühendisliği: Teoriden Pratiğe*, 1. baskı, ODTÜ Geliştirme Vakfı Yayıncılık, Ankara, 2011.

Darnell E., Halgren S., *Usability Throughout the Product Development Cycle*, 1st ed., Usability Professionals’ Association, Chicago, IL, 2008.

Da Silva T. S., Silveira M. S., Maurer F., Ten Lessons Learned from Integrating Interaction Design and Agile Development. *Proceedings - AGILE 2013*, DOI:10.1109/AGILE.2013.11.

Detweiler M., Managing UCD Within Agile Projects, *Interactions*, 2007, **14**(3), 40-42.

Dillon A., Sweeney M., Maguire M., *A Survey of Usability Engineering within the European IT industry-Current Practice and Needs*, 5th International Conference on Human-Computer Interaction, Orlando, Florida, USA, 24-29 April 1993.

Faulkner C., *The Essence of Human-Computer Interaction*, 1st ed., Prentice Hall, London, England, 1998.

Ferre X., *Integration of Usability Techniques into the Software Development Process*, 25th ICSE, Portland, Oregon, USA, 3-10 May 2003.

Ferreira J., Noble J., Biddle R., *Agile Development Iterations and UI Design, Proceedings - AGILE 2007*, DOI:10.1109/AGILE.2007.8.

Gould J. D., Lewis C., *Designing for Usability: Key Principles and What Designers Think*, *Communications of the ACM*, 1985, **28**(3), 300-311.

Göransson B., Gulliksen J., Boivie I., *The Usability Design Process—Integrating User-Centered Systems Design in the Software Development Process*, *Research Section*, DOI:10.1002/spip.174.

Göransson B., Lif M., Gulliksen J., *Usability Design-Extending Rational Unified Process with a New Discipline*, 8th International Workshop on Interactive Systems, Funchal, Madeira Island, Portugal, 2003.

Green P., Wei-Haas L., *The Rapid Development of User Interfaces: Experience with the Wizard of Oz Method*, *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 1985, **29**(5), 470-474.

Gulliksen J., Boivie I., Persson J., Hektor A., Herulf L., *Making a Difference: A Survey of the Usability Profession in Sweden*, *Proceedings of the third Nordic Conference on Human-Computer Interaction*, DOI: 10.1145/1028014.1028046.

Hartson H.R., Hix D., *Toward Empirically Derived Methodologies and Tools for Human Computer Interface Development*. *International Journal of Man-Machine Studies*, 1989, **31**(4), 477-494.

Hellmann T. D., Hosseini-Khayat A., Maurer F., *Supporting Test-Driven Development of Graphical User Interfaces Using Agile Interaction Design*, 3rd International Conference on Software Testing, Verification, and Validation Workshop, Paris, France, 2010.

Hix D., Hartson H. R., *Developing User Interfaces: Ensuring Usability Through Product and Process*, 1st ed., John Wiley & Sons, Inc, New York, 1993.

Holzinger A., Errath M., Searle G., Thurnher B., Slany W., *From Extreme Programming and Usability Engineering to Extreme Usability in Software Engineering Education (XP+UE→XU)*, The 29th Annual International Computer Software and Applications Conference, Edinburgh, Scotland, 2005.

Horn J. A., *Evaluation of Behavior Driven Development*, Doctoral Dissertation, TUDelft:Delft University of Technology, Delft, 2012.

Hsieh H. F., Shannon S. E., Three Approaches to Qualitative Content Analysis, *Qualitative Health Research*, 2005, **15**(9), 1277-1288.

Humayoun S. R., Dubinsky Y., Catarci T., A Three-Fold Integration Framework to Incorporate User – Centered Design into Agile Software Development, *Human Centered Design*, DOI: 10.1177/1049732305276687.

Hussain Z., Slany W., Holzinger A., Current State of Agile User-Centered Design: A Survey, *Symposium of the Austrian HCI and Usability Engineering Group*, Linz, Austria, 9 November 2009.

ISO13407, 13407: Human-centred design processes for interactive systems, *ISO*, Geneva, 1999.

Janzen D., Saiedian H., Test-Driven Development: Concepts, Taxonomy, and Future Direction, *Computer*, 2005, **38**(9), 43-50.

Highsmith J., *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, 1st ed., Addison-Wesley, New York, 2013.

Ji Y. G., Yun M. H., Enhancing the Minority Discipline in the IT Industry: A Survey of Usability and User-Centered Design Practice, *International Journal of Human-Computer Interaction*, 2006, **20**(2), 117-134.

Kruchten P., *The Rational Unified Process: An Introduction*, Reading, Massachusetts, 2nd ed., Addison Wesley Professional, Boston, 2004.

Lee J. C., McCrickard S., Stevens D.S., Examining the Foundations of Agile Usability with Extreme Scenario-Based Design, *Agile Conference 2009*, Chicago, USA, 24-28 August 2009.

Losada B., Urretavizcaya M., FernándezCastro I., A Guide to Agile Development of Interactive Software with a “User Objectives” Driven Methodology, *Science of Computer Programming*, DOI:10.1016/j.scico.2012.07.022

Losada B., Urretavizcaya M., López-Gil J.-M., Fernández-Castro I., Combining InterMod Agile Methodology with Usability Engineering in a Mobile Application Development, *Proceedings of the 13th International Conference on Interacción Persona-Ordenador - INTERACCION '12*, DOI:10.1145/2379636.2379674

Mayhew D. J., *The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design*, 1st ed., Morgan Kaufmann, San Francisco, 1999.

Memmel T., Box D., Box D., Reiterer H., Box D., Agile Human-Centered Software Engineering, *21st British HCI Group Annual Conference on People and Computers*, Lancaster, United Kingdom, 3-7 September 2007.

Memmel T., Gundelsweiler F., Reiterer H., CRUISER: A Cross-Discipline User Interface and Software Engineering Lifecycle. *12th International Conference on Human-Computer Interaction*, Beijing, P.R. China, 22-27 July 2007.

Natalia V. I., John V. C., Sheldon L.S., Using Mixed Methods Sequential Explanatory Design: From Theory to Practise, *Field Methods*, DOI: 10.1177/1525822X05282260.

Nielsen J., Usability Inspection Methods, *Proceeding CHI '94 Conference Companion on Human Factors in Computing Systems*, DOI: 10.1145/259963.260531.

Roberts P., Priest H., Reliability and Validity in Research, *Nursing Standard*, 2006, **20**(4), 41-45.

Rolf G., Validity, Trustworthiness and Rigour; Quality and Idea or Qualitative Research, *Journal of Advanced Nursing*, 2006, **53**(3), 304-310.

Rosenbaum S., Rohn, J. A., Humburg J., A Toolkit for Strategic Usability: Results from Workshops, Panels, and Surveys, *SIGCHI Conference on Human Factors in Computing Systems*, The Hague, Netherlands, 01-06 April 2000.

Rosson M. B., Carroll J. M., *Usability Engineering, Scenario-Based Development of Human-Computer Interaction*, 1st ed., Morgan Kaufmann Publishers, San Francisco, 2001.

Smart J. F., *BDD in Action.*, 1st ed., Manning Publications, USA, 2014.

Seffah A., Metzker E., The Obstacles and Myths of Usability and Software Engineering, *Communications of the ACM*, 2004, **47**(12), 71-76.

Solis C., Wang X., A Study of the Characteristics of Behaviour Driven Development, *37th EUROMICRO Conference on Software Engineering and Advanced Applications*, Oulu, Finland, 30 August – 02 September 2011.

Sommerville I., *Software Engineering*, 9th ed., Pearson, Boston, 2010.

URL-1: Altıntaş A., *BDD*, <https://www.youtube.com/watch?v=7GpnZXmrc2I> (Ziyaret Tarihi:30 Mart 2015)

URL-2: Azam M., *Behavior-Driven Development Using SpecFlow*, <http://www.codemag.com/Article/1208081>, (Ziyaret Tarihi:30 Mart 2015).

URL-3: BLM 518 2014-2015 Yüksek Lisans Dersi, <http://akademikpersonel.kocaeli.edu.tr/pinar.onaydurdu/index.php?a=Akademisyen&bilgi=ders>, (Ziyaret Tarihi:12 Mayıs 2016).

URL-4: Esquivel G., *Differences Between TDD, ATDD and BDD*, <http://gaboesquivel.com/blog/2014/differences-between-tdd-atdd-and-bdd/>, (Ziyaret Tarihi:12 Mayıs 2016).

URL-5: Eargle C., *JustCode and Jasmine (BDD JavaScript testing framework)*, <http://blogs.telerik.com/justteam/posts/12-02-24/justcode-and-jasmine-bdd-javascript-testing-framework-play-together.aspx>, (Ziyaret Tarihi:30 Mart 2015).

URL-6: Gunther R., Janis J., Butler S., *The UCD Decision Matrix: How, When, and Where to Sell User-Centered Design into the Development Cycle*, <http://www.ovostudios.com/upa2001/>, (Ziyaret Tarihi: 12 Nisan 2016).

URL-7: <http://botemlk.blogspot.com.tr/2009/05/kullanlabilirlik-testi.html>, (Ziyaret Tarihi: 30 Mart 2015).

URL-8: North D., *Introducing Bdd*, <https://dannorth.net/introducing-bdd/>, (Ziyaret Tarihi: 5 Nisan 2015).

URL-9: Peham T., *5 Types Of User Acceptance Tests – The Perfect Uat Framework*, <http://usersnap.com/blog/types-user-acceptance-tests-frameworks/>, (Ziyaret Tarihi: 30 Mart 2015).

URL-10: <http://www.agilemanifesto.org/principles.html>, (Ziyaret Tarihi: 30 Mart 2015)

URL-11: Shubin H., Details of a Usability Study, *Interaction Design Inc*, <http://www.user.com/testing-details.htm>, (Ziyaret Tarihi: 10 Nisan 2015).

URL-12: <https://www.techopedia.com/definition/3887/user-acceptance-testing-uat>, (Ziyaret Tarihi: 30 Mart 2015).

Venturi G., Troost J., Survey on the UCD Integration in the Industry, *Proceedings of the Third Nordic Conference on Human-Computer Interaction*, DOI: 10.1145/1028014.1028092

Vredenburg K., Mao J. Y., Smith P. W., Carey T., A Survey of User-Centered Design Practice. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, DOI: 10.1145/503376.503460.

Wynne M., Hellesoy A., *The Cucumber Book: Behaviour-Driven Development for Testers and Developers*, 1st ed., Pragmatic Bookshelf, USA, 2012.

Yıldırım K., Nitel Araştırmalarda Niteliği Artırma, *İlköğretim Online*, 2010, **9**(1), 79-92.

Zhang P., Carey J., Te'eni D., Tremaine M., Integrating Human-Computer Interaction Development into the Systems Development Life Cycle: A Methodology, *Communications of the Association for Information Systems*, 2005, **15**(1), 29.

Zhou R., Huang S., Qin X., Huang S., A Survey of User-Centered Design Practice in China, *Systems, Man and Cybernetics 2008 IEE International Conference*, Singapore, 12-15 October 2008.



EKLER

EK-A ANKET SORULARI

Kullanılabilirliğin Yazılım Yaşam Döngüsüne Entegrasyonu Anket Çalışması

(* İşareti ile başlayan sorular zorunludur.)

1.e-posta adresiniz:

_____@_____

2.*Cinsiyetiniz:

(Sadece bir seçeneği işaretleyiniz.)

Bayan

☐

Bay

☐

3.Yaşınız:

4.*Eğitim durumunuz:

(Sadece bir seçeneği işaretleyiniz.)

Doktora

☐

Lisans

☐

Lise

☐

Yüksek Lisans

☐

Ön lisans

☐

5.Lisans derecenizi aldığınız alan?

	Bölümler
Mühendislik Fakültesi	_____
Fen-Eğitim Fakültesi	_____
Beşeri Bilimler	_____
Sosyal Bilimler	_____
Davranış Bilimleri	_____
Diğer	_____

6.*Şirketiniz hangi alanda faaliyet göstermektedir? Size en uygun olan alını seçiniz.

(Sadece bir seçeneği işaretleyiniz.)

Havacılık

☐

Otomotiv

☐

Bilgisayar

☐

Eğitim

☐

Finans Hizmetleri	☐
Araştırma	☐
Sağlık hizmetleri	☐
İnternet / E-Ticaret	☐
İBE/Kullanılabilirlik Danışmanlığı	☐
İmalat	☐
Yayıncılık	☐
Telekomünikasyon	☐
Toptan/Perakende	☐
Devlet	☐
Metin	☐ _____

7.*Şirketinizde yaklaşık kaç personel çalışmaktadır?

8.*Çalışmakta olduğunuz şirket alanında yaklaşık kaç yıldır faaliyet göstermektedir:

9.*Şirket içerisindeki unvanınız?

(Sadece bir seçeneği işaretleyiniz.)

Kullanılabilirlik uzmanı/mimarı/tasarımcısı/mühendisi	☐
Etkileşim tasarımcısı/mimarı	☐
Sahip, yönetici, denetimsel pozisyon	☐
BT danışmanı, danışman	☐
Yazılım/sistem tasarımcısı/geliştirici, programcı	☐
Web tasarımcısı/master/editörü	☐
Sistem yöneticisi/mühendis/mimarı/	☐
Tasarımcı/Arayüz tasarımcısı	☐
BT stratejisti/çalışanı/mühendisi	☐
İş analisti/geliştiricisi	☐
Kullanıcı deneyimi analisti/tasarımcısı/yöneticisi	☐
Diğer	☐ _____

10.*Şirketinizde yazılım geliştirme projelerinde kullandığınız resmi bir Yazılım Geliştirme Yöntemi uygulanmakta mıdır?

(Sadece bir seçeneği işaretleyiniz.)

- Evet ☐
- Hayır, bazı araç ve teknikleri kullanıyoruz
fakat standart bir yöntemimiz yok ☐
- Hayır, bir standardımız yok her projede farklı
yaklaşımları uyguluyoruz ☐
- Diğer ☐ _____

11.*Şirketinizde kullanılan Yazılım Geliştirme Yöntemi aşağıdakilerden hangisine en çok uymaktadır?

(Sadece bir seçeneği işaretleyiniz.)

- Şirket içinde geliştirdiğimiz kendimize ait bir
yöntem ☐
- Rational Unified Process (RUP) ☐
- Agile Unified Process ☐
- Çağlayan (waterfall) ☐
- SCRUM ☐
- Extreme Programming (XP) ☐
- Lean Development ☐
- Crytal ☐
- Diğer ☐ _____

12.*Yazılım projelerini geliştirirken, iş bölümünü yazılım ekibi içerisindeki farklı gruplarla mı yönetiyorsunuz?(back-end front-end developing)

(Sadece bir seçeneği işaretleyiniz.)

- Evet ☐
- Hayır ☐
- Proje bazlı değişim göstermektedir. ☐
- Diğer ☐ _____

13.*Yazılım geliştirme sırasında en çok test ettiğiniz işleme en yüksek puanı vererek sıralayınız.

(Önceliklerine göre sıralayınız.)

	1	2	3	4
Fonksiyonalite	☐	☐	☐	☐
Kullanılabilirlik	☐	☐	☐	☐
Görsel tasarım	☐	☐	☐	☐
Güvenilirlik	☐	☐	☐	☐

14.*Aşağıdaki proje gereksinimlerini önceliğine göre sıralar mısınız?(En

öncelikli olanı ilk sıraya koyunuz.)

(Önceliklerine göre sıralayınız.)

	1	2	3	4
Fonksiyonalite	☐	☐	☐	☐
Kullanılabilirlik	☐	☐	☐	☐
Görsel tasarım	☐	☐	☐	☐
Güvenilirlik	☐	☐	☐	☐

15.*Yazılım projesi bittiğinde, geliştirme öncesinde belirlediğiniz hedefleri gerçekleştirme başarınıza göre puanlandırınız.

(Önceliklerine göre sıralayınız.)

	1	2	3	4
Fonksiyonalite	☐	☐	☐	☐
Kullanılabilirlik	☐	☐	☐	☐
Görsel tasarım	☐	☐	☐	☐
Güvenilirlik	☐	☐	☐	☐

16.*Geliştirdiğiniz projelerde kullanılabilirliğin sağlanması önemli bir konudur?

(Sadece bir seçeneği işaretleyiniz.)

- Evet ☐
- Bazen, projesine göre değişir ☐
- Hayır ☐

17.*Kullanılabilirliği proje gereksinimlerinize hangi durumlarda dahil edersiniz?

(Bir ya da birden çok seçim yapabilirsiniz.)

- Her zaman ☐
- Sadece projede kullanılabilirlikle ilgili bir problem çıktığında ☐
- Sadece müşteri talep ederse ☐
- Sadece projede bir kullanılabilirlik uzmanı bulundurulabilirsek ☐
- Diğer ☐ _____

18.*Kullanılabilirlikle ilgili gereksinimleri nasıl belirlersiniz?

(Bir ya da birden çok seçim yapabilirsiniz.)

- Kullanıcılarla görüşmeler yaparak ☐

- Daha önceki projelerdeki iyi deneyimlerimize bakarak ☐
- Kitaplar ya da Internetten araştırarak ☐
- Diğer ☐ _____

19.*Şirketinizde yazılım geliştirme sırasında kullanılabilirliğin sağlanmasına yönelik aktiviteler (kullanılabilirlik testleri, buluşsal değerlendirme, vb) gerçekleştiriliyor mu?

(Sadece bir seçeneği işaretleyiniz.)

- Evet ☐
- Hayır ☐

19. soruya EVET cevabı verenler 20-28 arasındaki soruları yanıtlayacaklardır 19. soruya HAYIR cevabı verenler 29. sorudan sonraki soruları yanıtlayacaklardır

20.Yaklaşık ne kadar süredir bu aktiviteler gerçekleştiriliyor?

21.Şirketinizde İBE(İnsan-Bilgisayar Etkileşimi) ya da kullanılabilirlikle ilgili herhangi bir eğitime sahip personeliniz var mı?

(Sadece bir seçeneği işaretleyiniz.)

- Evet ☐
- Hayır ☐
- Bilmiyorum ☐

22.Şirketinizde İBE ya da kullanılabilirlikle ilgili eğitime sahip personelin sahip oldukları bilgi geçmişi nedir?

(Bir ya da birden çok seçim yapabilirsiniz.)

- İBE ve kullanılabilirlik konusunda bir programa (derece) devam ederek bilgi sahibi olmuşlar ☐
- İBE ile ilgili bir ders ya da programa devam ederek bilgi sahibi olmuşlar ☐
- Kendi kendilerine öğrenmişler ☐
- İş başında çalışarak öğrenmişler ☐
- Bilmiyorum ☐
- Diğer ☐ _____

23.Şirketinizde aşağıdaki ünvanlara sahip kullanılabilirlik konusundan sorumlu personel bulunmakta mıdır? Birden fazla işaretleyebilirsiniz

(Bir ya da birden çok seçim yapabilirsiniz.)

Kullanıcı Deneyimi Tasarımcısı	☐
Etkileşim Tasarımcısı	☐
Kullanıcı Deneyimi Tasarımcısı	☐
Kullanıcı Deneyimi Yöneticisi	☐
Kullanıcı Araştırmacısı	☐
Bilgi Mimarı	☐
Kullanılabilirlik Profesyoneli	☐
Kullanılabilirlik Mühendisi	☐
Arayüz Tasarımcısı	☐
Ürün Tasarımcısı	☐
Kullanıcı Deneyimi Mimarı	☐
Kullanılabilirlik Uzmanı	☐
Diğer:	☐ _____

24.Şirketinizde kullanılabilirliğin sağlanmasına yönelik aşağıdaki yöntemler hangi sıklıkla kullanılmaktadır?

(Her satırda sadece bir seçeneği işaretleyiniz.)

	Hiç ir zama n	Nadir en	Sıkç a	Genellik le	Herzaman	Bilmiyorum
Kullanılabilirlik ile ilgili plug- in kullanmak	☐	☐	☐	☐	☐	☐
ISO 13407 gibi genel kullanılabilirlik çerçevelerini kullanmak	☐	☐	☐	☐	☐	☐
Genel İBE yöntemlerini kullanmak (Kullanıcı çalışmaları, kullanıcı testleri, alan araştırmaları, vb)	☐	☐	☐	☐	☐	☐
Buluşsal değerlendirme gibi uzman tabanlı yöntemleri kullanmak (	☐	☐	☐	☐	☐	☐

Nielsen'in kullanılabilirlik buluşsalları gibi)						
Stil Kılavuzlarına dayalı değerlendirme yapmak (Apple Tasarım Prensipleri gibi)	☐	☐	☐	☐	☐	☐

25.Kullanılabilirlik testlerinde sistemi değerlendirecek kullanıcılarınızı nasıl belirliyorsunuz?

(Bir ya da birden çok seçim yapabilirsiniz.)

Gelişmiş kullanıcılar test ettiriyoruz ☐

Sistemin hedef kullanıcı grubundan temsili bir örnekleme test ettiriyoruz ☐

Kendi çalışanlarımıza test ettiriyoruz ☐

Müşterimizin çalışanlarına test ettiriyoruz ☐

Kullanılabilirlik testi yapmıyoruz ☐

Diğer ☐ _____

26.Kullanılabilirlik testlerinizi tipik olarak kaç kullanıcı ile gerçekleştiriyorsunuz?

27.Kullanılabilirlik ile ilgili verilen yöntem ve teknikleri belirtilen hangi evrelerde kullandığınız işaretleyiniz.

(Her satırda sadece bir seçeneği işaretleyiniz.)

	Ön- çalışma (fizibilite)	Gereksini m	Tasarım	Yapım(Kodlama)	Test	Bakı m
Kullanım Durumu (Use Case)	☐	☐	☐	☐	☐	☐
Storyboard	☐	☐	☐	☐	☐	☐
Vizyon/kapsam dokümanı	☐	☐	☐	☐	☐	☐

Grafik Arayüz Spesifikasyonlar 1	☐	☐	☐	☐	☐	☐
Kağıt ya da yazılım prototipler	☐	☐	☐	☐	☐	☐
Müşteri görüşmeleri	☐	☐	☐	☐	☐	☐
Kullanılabilirlik testleri	☐	☐	☐	☐	☐	☐
Buluşsal (heuristic) değerlendirme	☐	☐	☐	☐	☐	☐
Kontrol listeleri	☐	☐	☐	☐	☐	☐
Stil Kılavuzları	☐	☐	☐	☐	☐	☐
Görev analizi	☐	☐	☐	☐	☐	☐
Memnuniyet anketleri	☐	☐	☐	☐	☐	☐
Persona/kullanıcı profilleri	☐	☐	☐	☐	☐	☐
Kıyaslama (benchmarking) çalışmaları	☐	☐	☐	☐	☐	☐
Görsel tasarım	☐	☐	☐	☐	☐	☐
Kullanım loglarının analizi	☐	☐	☐	☐	☐	☐

28.Kullanıcıları Yazılım geliştirme sürecinizin hangi aşamalarına dahil ediyorsunuz? Birden fazla seçebilirsiniz.

(Bir ya da birden çok seçim yapabilirsiniz.)

- Ön-çalışma ☐
- Gereksinim analizi ☐
- Tasarım fazı ☐
- Yapım fazı ☐
- Test fazı ☐
- Teslimat öncesi ☐
- Teslimat sonrası/bakım ☐

19.Soruya HAYIR cevabı verenler buradan sonra devam edebilirler

29.Şirketinizde yazılım geliştirme sırasında kullanılabilirliğin sağlanmasına yönelik aktivitelerin (kullanılabilirlik testler, buluşsal değerlendirme, vb) gerçekleştirilmemesi ile ilgili nedenler nelerdir? Birden fazla işaretleyebilirsiniz?

(Bir ya da birden çok seçim yapabilirsiniz.)

- Bütçe ayrılmaması ☐
- Zamanın yeterli olmaması ☐
- Proje ekibinin sayıca yetersiz olması ☐
- Yönetim ya da mühendislerin kullanılabilirliği gerekli görmemesi ☐
- Kullanılabilirliğin şirket içerisinde yeterince bilinmemesi ☐
- Ne faydasının olacağının hesaplanamaması ☐
- Konu ile ilgili uzman personelin bulunmaması ☐
- Ekonomik ihtiyacın olmaması ? müşterinin istememesi ☐

30.Yazılım projesine başlamadan önce aşağıdaki gereksinimleri şirketinizce öncelik sırasına değerlendiriniz

(Her satırda sadece bir seçeneği işaretleyiniz.)

	Kesinlikle katılmıyorum	Katılmıyorum	Kararsızım	Katılıyorum	Kesinlikle katılıyorum
Kullanılabilirlikle ilgili gereksinimler projelerin başarısı için çok önemlidir	☐	☐	☐	☐	☐
Kullanılabilirlik başlangıçtan itibaren proje planının bir parçasıdır	☐	☐	☐	☐	☐
Kullanılabilirlik proje geliştirimi sırasında zaman kaybıdır.	☐	☐	☐	☐	☐
Kullanılabilirlik şirketlere extra maliyet yüklemektedir.	☐	☐	☐	☐	☐

Proje geliştirimi sırasında kullanılabilirlik açısından kullanıcıların sürece dahil olması önemlidir.	☐	☐	☐	☐	☐
Kullanılabilirlik testleri projenin başarısı için çok önemlidir	☐	☐	☐	☐	☐
Kullanılabilirliğin sağlanmasına yönelik aktiviteler şirket içerisinde kullandığımız Yazılım Geliştirme Yöntemine iyi entegre olmuştur	☐	☐	☐	☐	☐
Şirketimizde kullanılabilirlik konusu ile ilgili farkındalık yeterince vardır	☐	☐	☐	☐	☐
Şirketimizde kullanılabilirlik le ilgili aktiviteler üst yönetim tarafından da desteklenmektedir	☐	☐	☐	☐	☐
Kullanılabilirliğin sağlanması projelerin maliyetini	☐	☐	☐	☐	☐

arttırıcı bir unsurdur					
Kullanılabilirlik ile ilgili aktivitelerin gerçekleştirme için şirket içerisinde konu ile ilgili resmi bir eğitim almış kişilerin bulunması gereklidir	☐	☐	☐	☐	☐
Kullanılabilirlik ile ilgili aktivitelerin çevik yöntemlere daha uygun olduğunu düşünüyorum	☐	☐	☐	☐	☐

31.Anket sonuçlarıyla ilgili bilgi almak istiyorsanız iletişime geçmemizi istediğiniz e-posta adresini belirtiniz:

_____@_____

EK-B GÖRÜŞME SORULARI

A. Katılımcılar, organizasyonları ve projeler ile ilgili bilgiler

1. Kısaca kendinizi tanıtabilir misiniz?
 - a. Organizasyon içerisindeki göreviniz (ünvan) / sorumluluklarınız?
 - b. Ne kadar süredir bu oragnizasyonda görev alıyorsunuz?
2. Şirket bilgileriniz nelerdir?
 - a. Organizasyonunuzun büyüklüğü- çalışan sayınız vb
 - b. Ne kadar zamandır faaliyet gösteriyorsunuz?
 - c. Hangi sektörlere yönelik projeler geliştiriyorsunuz?
 - i. Geliştirilen proje türleri?
 - ii. Bu projelerde kullanıcı etkileşimi ne kadar ön planda?
3. İBE (İnsan Bilgisayar Etkileşimi) ya da kullanılabilirliğin sağlanmasıyla ilgili/sorumlu herhangi bir eğitime sahip personeliniz var mı?
 - a. Hangi ünvanla anılıyor?
 - b. Kişinin eğitimi nereden geliyor
(Diploma programı, sertifika programı, ilgili bir ders alma, iş başında deneyimle)
 - c. İlgili ünvana sahip bir personel yoksa bu sorumluluğu kim nasıl gerçekleştiriyor

B. Kullanılabilirlik kavramının anlaşılması ve uygulanması

4. Kullanılabilirliğin kısaca tanımını yapabilir misiniz?
5. Projelerinizde genellikle sayacağım konulardan hangileri daha önceliklidir;
 - a. Kullanılabilirlik
 - b. Fonksiyonalite
 - c. Görsel tasarım
 - d. Güvenilirlik
6. En çok test edilen gereksinim türü bu saydıklarımızdan hangileridir?
 - a.Kullanılabilirlik b. Fonksiyonalite c.Görsel tasarım d. Güvenilirlik
7. Kullanılabilirliğin geliştirme projelerinde önemi nedir? Ya da şirketinizde geliştirilen projelerde 'kullanılabilirlik' konusuna ne derece önem verilmektedir?
 - a. Sizce kurum içerisinde bu konuda yeterli düzeyde farkındalık var mıdır?
8. Yazılım projesi bittiğinde, geliştirme öncesinde belirlediğiniz hedefleri gereksinim türüne göre gerçekleştirme başarınızdan bahsedebilir misiniz? Yani projelerde genellikle hangi tür gereksinimleri karşılamada daha başarılı olduğunuzu düşünüyorsunuz?
 - a. Kullanılabilirlik b.Fonksiyonalite c. Görsel tasarım d. Güvenilirlik
9. Yazılım projelerinizi geliştirirken belli bir Yazılım Yaşam Döngüsü Modeli kullanıyor musunuz?
 - a. Hangi yazılım geliştirme metodunu kullanıyorsunuz.
Örnek: Çağlayan (waterfall), RUP, XP gibi bir çevik yöntem vb.
 - i. Bu yaşam döngüsü modelini nasıl uyguladığınızı açıklayabilir misiniz?

- b. Eğer bir isimlendirme yapılamıyorsa: Kendi yazılım sürecinizi ana hatlarıyla açıkla mısınız?
- 10. Herhangi bir modelleme yöntemi (UML gibi) kullanıyor musunuz?
 - a. Kullanıyorsanız; kullanıyor olmaktan memnun musunuz?
 - b. Kullanmıyorsanız; bir eksikliğini hissediyor musunuz?
 - i. Bir eksiklik hissetmiyorsanız bunu neye bağlıyorsunuz?
- 11. Kullanılabilirliği uyguladığınız yaşam döngüsü modelinin neresinde görüyorsunuz?
 - a. Hangi aşama ya da aşamalarda uygulanması gerektiğini düşünüyorsunuz? (ön çalışma, gereksinim, tasarım, yapım, test veya bakım)
 - b. Uygulamak isteyip de uygulayamadığınız aşamalar var mı?
 - c. Uygulayabildiğiniz aşamalardaki uygulama yaklaşımlarınızı/sürecinizi açıklayabilir misiniz?

C. Kullanılabilirlik değerlendirmelerinin gerçekleştirilmesi

- 12. Kullanılabilirlik ile ilgili gereksinimlerini nasıl topluyorsunuz?
 - a. Bu işlemleri yapacak uzman personeliniz/ ilgili birim var mı?
 - b. Ya da bu gereksinimleri toplamak üzere bir birim tanımlanmış durumda mı?
 - c. Gereksinimleri tespit etmek için müşterilerinizden yararlanıyor musunuz?
- 13. Kullanılabilirliği test etmede hangi yöntem ve teknikleri kullanıyorsunuz?(Örn. Stil kalvuzları, kullanıcı testleri, sesli düşünme, vb)
- 14. Gerçek kullanıcıları gereksinim toplama ya da testleri sırasında kullanıyor musunuz?
 - a. Kullanılabilirlik değerlendirmesini kimler ile gerçekleştiriyorsunuz?
 - b. Ortalama kaç katılımcı ile gerçekleştiriyorsunuz?
 - c. Sizce gerçek kullanıcıların sürece dahil olmasının faydaları nelerdir?

- 15. Kullanılabilirlik değerlendirme sonuçlarınızı sürecin hangi aşamasında kimlere geribildirim sağlamada kullanıyorsunuz?

D. Yazılım geliştirme profesyonellerinin kullanılabilirlik alguları

- 16. Kullanılabilirlik değerlendirmesinin yazılım geliştirme sürecinde uygulanmasının avantajları ve dezavantajları sizce nelerdir?
- 17. Kullandığınız yazılım yaşam döngüsü modeline kullanılabilirliğin ne derecede entegre edildiğini düşünüyorsunuz?
 - a. Kullanılabilirlikle ilgili yöntemlerin etkili bir şekilde projelerde uygulanamama nedenleri sizce neler olabilir?
 - b. Kullanılabilirliğin daha etkili uygulanabilmesi için gerekli anahtar faktörler sizce neler olabilir?
 - i. Sizin uygulamalarınızdan edindiğiniz pratik öneriler nelerdir
- 18. Projelerinizde kullandığınız yazılım geliştirme sürecine kullanılabilirliğin sağlanması için gerekli olan aktiviteler daha etkili olarak nasıl entegre edilebilir?

EK-C MODEL DEĞERLENDİRMESİ GÖRÜŞME SORULARI

A. Katılımcılar, organizasyonları

19. Kısaca kendinizi tanıtabilir misiniz?

- Daha önce çalıştığınız/ Şuan çalışmakta olduğunuz organizasyonda göreviniz nedir?
- Ne kadar süredir bu görevde yer alıyorsunuz?
- Görev aldığınız şirket ya da kurum hangi sektörde hizmet vermektedir?

B. Kullanıcı deneyimi konusunda Tecrübeler

2. Kullanıcı deneyimi tasarımı tecrübeniz hakkında bilgi verir misiniz?

- Daha önce çalıştığınız/ görev aldığınız projelerde kullanıcının yeri ve öneminden bahseder misiniz?
- Kullanıcı deneyimine önem veren tasarım projelerinde görev aldınız mı? Aldıysanız eğer kullanıcı odaklı bir proje üretmek için kullandığınız özel yöntemler var mıydı?

C. Çevik Yöntemlerin Kullanımı Konusunda Tecrube

3. Çevik yöntemler hakkında bilgi sahibi misiniz?

- Çevik yöntemleri kullandığınız bir projede yer aldınız mı?
- Çevik yöntemlerden
 - TDD
 - BDD hakkında bilgi sahibi misiniz? Uygulamalarınızda bu yöntemleri kullandınız mı?
 - Eğer bu yöntemleri kullandıysanız BDD formatını kullanırken zorlandınız mı?

D. Ders projesinde uygulanan BUSDM Analiz Safhası

4. BDD kullanımı

- Ders kapsamında gerçekleştirdiğiniz projenizi açıkla mısınız? Nasıl bir süreç ve takım çalışması takip ettiniz? Analizden sona kadar
- BDD yazımı format açısından kolay bir yaklaşım mı? Kurguladığınız senaryoları DOG(BDD) formatıyla yazarken zorlandığınız oldu mu?
- Ders projesinde personaları oluşturduktan sonra KOD senaryoları yazıldı. Çıkarılan personaların KOD senaryo yazımında katkısı oldu mu? Eğer olduysa, senaryo yazımında personaları nasıl entegre ettiniz?
- KOD senaryosu yazımından sonra prototipleme adımına geçildi. Yazılan KOD senaryoları sizlere prototiplemede yardımcı oldu mu?

E. BDD ve Kullanıcı Odaklı Tasarım

5. KOD senaryolarıyla analizleri yazılan bir sistemin kullanıcı merkezli tasarıma uygunluk derecesi nedir? Katkısı konusunda görüşleriniz nelerdir?

EK-D GÖRÜŞME SORULARINA VERİLEN CEVAPLAR

Tablo D.1

Kategori no	Kategori adı	Katılımcı kodu	Cevaplar
KG_1_1	İBE ya da kullanılabilirlik konularıyla ilgili personel	K1_YG_1	Yok. Test mühendisleri
		K1_YG_2	Var. Uzman/Uzman Yardımcısı kadrosunda(Arayüz geliştirmesi yapmış yazılımcı)
		K4_KU_1	Var. İş analisti
		K3_İA_1	Yok. Kullanılabilirliğe ilişkin bir çalışma gerçekleştirilmiyor. İş analistleri projelerde kullanılabilirliği dikkate almaya çalışarak analiz safhasını yürütüyor.
		K2_YG_3	Kullanıcıların ihtiyaçlarını, kullanılabilirliğini belirleyen uygulamaların, kullanıcıların ihtiyaçlarına göre hazırlayan analistler var.
		K5_TM_1	Yok. Test mühendisleri
		K3_UY_1	Yok. Analist ve yazılımcılar birlikte bu sorumluluğu paylaşmakta.
		K6_TM_2	Yok. Product & Testing Departmanı / Product Engineer mühendisleri bu görevi üstlenmektedir.
KG_1_2	Kullanılabilirliğin sağlanmasına verilen önem	K1_YG_1	Projelerimizde genellikle ; <u>güvenilirlik ve fonksiyonlité</u> daha önceliklidir. En çok onlar test edilir. Proje bitiminde <u>güvenilirlik</u> konusundaki başarı diğerlerine göre daha yüksektir. Kullanılabilirliğe gereken önem verilmemektedir.
		K1_YG_2	Projelerimizde genellikle <u>fonksiyonlité</u> daha önceliklidir. En çok o test edilir. Proje bitiminde fonksiyonlité konusundaki başarı diğerlerine göre daha yüksektir. Kurum içerisinde bu konuda yeterli düzeyde farkındalık yok.
		K4_KU_1	Projelerimizde genellikle <u>kullanılabilirlik ve fonksiyonlité</u> daha önceliklidir. En çok <u>fonksiyonlité</u> test

			edilir. Proje bitiminde <u>fonksiyonallite</u> konusundaki başarı diğerlerine göre daha yüksektir. BT içerisinde üst seviyelerde konu öncelikli olarak değerlendirilmekte ve tüm organizasyon buna göre çalışmalarını sürdürmekte.
		K3_İA_1	Projelerimizde genellikle <u>fonksiyonallite ve güvenilirlik</u> daha önceliklidir. En çok onlar test edilir. Proje bitiminde <u>fonksiyonallite ve güvenilirlik</u> konusundaki başarı diğerlerine göre daha yüksektir. Firmadaki projelerde kullanılabilirlik konusu için kaynak ayrılmamaktadır. Yeterli düzeyde farkındalık yok.
		K2_YG_3	Projelerimizde genellikle <u>kullanılabilirlik</u> daha önceliklidir. En çok <u>fonksiyonallite</u> test edilir. Proje bitiminde <u>fonksiyonallite</u> konusundaki başarı diğerlerine göre daha yüksektir. Önem veriliyor. Kullanıcılarımızı memnun etmek zorundayız. Projeleri sattığımız için de bunlara önem vermek zorundayız. Arka tarafta işi bi şekilde kotarıyoruz ama bunu kullanıcıya yansıtmadan onun istediğini göstermeye çalışıyoruz.
		K5_TM_1	Projelerimizde genellikle <u>fonksiyonallite</u> daha önceliklidir. En çok o test edilir. Proje bitiminde <u>fonksiyonallite</u> konusundaki başarı diğerlerine göre daha yüksektir. Bence yok genel olarak önem verilmiyor. Genellikle program çalışıyo mu çalışmıyo mu diye bakılıyor. Kullanılabilirlik arka planda kalıyor.
		K3_UY_1	Projelerimizde genellikle <u>güvenilirlik</u> daha önceliklidir. En çok <u>fonksiyonallite</u> test edilir. Proje bitiminde <u>güvenilirlik</u> konusundaki başarı diğerlerine göre daha yüksektir. Projelerdeki en önemli kriter değildir fakat fonksiyonalliteden sonra ikinci olarak en çok dikkat edilen noktadır. Yeterli olmasa da farkındalık var.
		K6_TM_2	Projelerimizde genellikle <u>fonksiyonallite</u> daha önceliklidir. Sonrasında <u>kullanılabilirlik</u> . En çok <u>fonksiyonallite</u>

			test edilir. Proje bitiminde fonksiyonalite konusundaki başarı diğerlerine göre daha yüksektir. Yeterince önem verilmektedir. Konu ile ilgili düzenli olarak toplantılar yapılmakta ve eğitimler alınmaktadır. Bölüm olarak yeterli düzeyde farkındalık var ancak kurum içinde malesef yok.
KG_1_3	Organizasyonlarda uygulanan yazılım yaşam döngüsü	K1_YG_1	Waterfall.
		K1_YG_2	Agile method.
		K4_KU_1	V-Model. Bir planlama fazıyla başlıyoruz. Analiz fazıyla devam ediyoruz. Analiz ve gereksinim onayları alıyoruz. Teknik tasarım ve yazılım fazıyla devam ediyor. Test i de kalite testi, fonksiyonel test, ön yüz standartları testi ve kullanıcı kabul testi adımlarına ayırıyoruz.
		K3_İA_1	Projelerin büyük çoğunluğunda waterfall kullanılıyor. Bazı projeler agile ile geliştiriliyor.
		K2_YG_3	Belirlenmiş bir yaşam döngüsü yok. Önce iş belirleniyor. Analistler tarafından süreçler belirlendikten sonra yazılımcılara geliyor iş. Bizden sonra teste gidiyor. Testten onaylandıktan sonra tekrar bize dönüyor. Biz onu canlıya çıkartıyoruz.
		K5_TM_1	Her aşamada test olan V model kullanıyoruz. Bazılarında agile kullandık. Şuan çalıştığımızda belli değil. Ama V modele uyuyor. Mesela biz developmenttan sonra test yapıyoruz. Analiz aşamasında da test oluyor. Daha sonra bizim testimiz bittikten sonra sistem testine gidiyor.
		K3_UY_1	Genellikle çağlayan yöntemi kullanılıyor, ama agile kullanılan projeler de olmakta.
		K6_TM_2	Kendi yöntemimiz kullanılıyor. Analysis > Analysis Review> Spec > Spec Review > Writing test codes > Coding > Spec Review > Refactoring >

			QA > UAT > Release
KG_1_4	Kullanılabilirliğin yaşam döngüsüne entegrasyonu	K1_YG_1	Kullanılabilirliğin <u>ön çalışmadan itibaren bütün adımlarda</u> olması gerektiğini düşünüyorum. Kullanılabilirlik bir yerde yapılacak olan projenin ne kadar fisible olup olmadığı ile alakalı.
		K1_YG_2	Kullanılabilirlik <u>tasarım ve önçalışmada yapılmalı</u> . Ama tasarımda hiç yapılmıyor.
		K4_KU_1	Kullanılabilirliğin <u>özellikle analiz ve testte olması gerektiğini</u> düşünüyorum bir de 'Kullanıcı Gözlem testi' çok zaman gerektirdiği için çoğu zaman atlanıyor. Bunu yapmak isterdik.
		K3_İA_1	Kullanılabilirlik <u>ön çalışma ve gereksinim fazlarında uygulanması gerekir</u> ; ancak hiçbir aşamada uygulanmıyor.
		K2_YG_3	<u>Kullanılabilirlik bizim için son aşamadaki test. Döngüde veri yok.</u>
		K5_TM_1	<u>Kullanılabilirlik bence analiz kısmında</u> başlaması gerekiyor ama her zaman öyle olmuyor. <u>Genelde bizde hata bulmak adına son aşamalarda</u> önemli oluyor.
		K3_UY_1	<u>Önçalışma ve tasarımda</u> yapılmalı. Ayrıca <u>ürünün son hali</u> üzerinden de testleri yapılmalıdır.
KG_1_5	Uygulanan modelleme yaklaşımları	K1_YG_1	Modelleme yaklaşımlarından ER (Entity relation diagrams) kullanıyoruz.
		K1_YG_2	Kullanıyoruz. UML
		K4_KU_1	Kullanmıyoruz. Eksikliğinin hissetmiyoruz. Pure Object Oriented Kod pratikte yazılamıyor.
		K3_İA_1	Hayır kullanılmıyor. Analiz süreçlerini hızlandırmak ve daha anlaşılabilir kılmak adına kullanılması gerekiyor.
		K2_YG_3	Kullanmıyoruz. Kesinlikle hissediyorum. Projenin okunurluğu yok. Mesela ben projeye sonradan dahil olduğum için eskiden yapılan işleri

			anlayabilmek için kod okumam gerekti. Bunun yerine şema çizilmiş olsaydı, db diyagramı olsaydı ilişkileri görmek adına daha kolay olurdu. İlişkileri anlayabilmek için tablolara tek tek gidip kendim çiziyorum diyagramları.
		K5_TM_1	<u>Hayır kullanmıyoruz.</u> Eksikliğini hissetmiyoruz şuanki projelerde.
		K3_UY_1	<u>Hayır kullanmıyoruz.</u> Eksikliği de hissedilmiyor.
		K6_TM_2	Moqups.com ile UI ve Spec'lere tanımladığımız <u>Interactions ve Use Case'ler</u> ile modelliyoruz. UML kullanmamak bir eksiklik değil çünkü çok formal.
KG_2_1	Kullanılabilirlik gereksinimlerinin toplanması	K1_YG_1	<u>Müşteri</u> ile yapılan görüşmeler sonucunda gereksinimleri topluyoruz. Bu görevi yapacak biri var mı şirkette bilmiyorum.
		K1_YG_2	<u>Anlaşma yapılan alıcı/kullanıcı kurumun istekleri</u> doğrultusunda gereksinimleri topluyoruz. Bu işlemleri yapacak personel var. Gereksinimleri toplarken müşterilerden yararlanıyoruz.
		K4_KU_1	Bazı standartlarımız var. <u>Kullanıcı ile yakın çalışıyoruz. Bu görevi iş analistleri üstleniyor.</u>
		K3_İA_1	Fonksiyonel gereksinimler için <u>müşterilerle/kullanıcılarla</u> iletişime geçiyoruz ancak kullanılabilirlik açısından herhangi bir etkileşim söz konusu değil. Bu görevi yapan uzman personel/birim yok.
		K2_YG_3	Gereksinimler <u>analistlerden</u> geliyor bize.
		K5_TM_1	Almanya tarafında toplanıyor. Sonra gereksinimler bize geliyor. Oradaki <u>analistlerle</u> toplanıyor.
		K3_UY_1	<u>Müşterilerden</u> yararlanıyoruz. Bu konuyla ilgili birim yok.
		K6_TM_2	Daha önceki benzer çalışmalarımıza, internetteki son trendlere ve rakiplerin yaptığı çalışmalara göz atarak topluyoruz. <u>Product ekibi</u> bu görevi

			üstleniyor.
KG_2_2	Kullanılabilirlik yöntemlerini uygulama	K1_YG_1	<u>Kullanıcı testleri. Müşteriler</u> ile gerçekleştiriyoruz. Ortalama <u>3-10</u> kişiyle.
		K1_YG_2	Müşteri ile toplantılar (<u>sesli düşünme</u>) <u>Gerçek kullanıcıları</u> değerlendirmeden kısmen kullanıyoruz. Bu değerlendirmeleri hem kurum içi test ekibi hem de müşteri test ekibi ile gerçekleştiriyoruz. <u>5 kişiyle.</u>
		K4_KU_1	<u>Kullanıcı testleri ve kural testleri (heuristic testing) Kullanıcılarla</u> gerçekleştiriyoruz. Kişi sayısı proje büyüklüğüne göre <u>değişiyor.</u>
		K3_İA_1	<u>Kullanılabilirliğe</u> ilişkin bir değerlendirme yapılmıyor.
		K2_YG_3	<u>Sadece kullanıcı testleri.</u> Analist test ediyor. Ben test ediyorum daha sonra sahadaki ekibe gönderilip, canlıda test edilmedi sağlanıyor. Sahadaki ekip: Uygulamayı gerçek kullanacak kişiler. Örneğin bizim lojistik projesindeyiz. <u>Müşterinin</u> de kendi <u>e-test ekibi</u> var. O da devreye almadan önce canlıya paralel şekilde kendi de test ediyor. Böylece <u>5-6 kişi</u> diyebiliriz.
		K5_TM_1	<u>Hiçbirini</u> kullanmıyoruz. Ama <u>Müşteriler</u> ile temas halinde oluyoruz bu aşamada.
		K3_UY_1	<u>Kullanıcı testleri, sesli düşünme. Test ekibi</u> ve <u>müşteriler</u> ile ortalama <u>5 testçi, 10 müşteri</u> ile gerçekleştiriyoruz.
		K6_TM_2	<u>Sesli düşünme</u> sırasında yorumları alıyoruz ve kullanıcı testleri yapıyoruz. <u>Stil kılavuzları</u> da gelecekte kullanacağız. <u>Product ekibi</u> ile <u>yaklaşık 5-10 kişiyle</u> gerçekleştiriyoruz.
KG_2_3	Kullanılabilirlik değerlendirme sonuçları	K1_YG_1	Değerlendirme sonuçlarını genelde proje bitiminde <u>müşterilere</u> iletiyoruz.
		K1_YG_2	Değerlendirme sonuçlarını <u>geliştirmecilere</u> geri bildirim yapmak amacıyla kullanıyoruz.
		K4_KU_1	Değerlendirme sonuçlarını BT de sorumlu <u>analiz ekiplerince</u> kullanıyor, problemleri noktaları düzeltiyor.

		K3_İA_1	
		K2_YG_3	Değerlendirme sonuçlarını <u>müşterilere</u> gösteriyoruz.
		K5_TM_1	Yazılım testi yaparken proje yöneticileriyle konuşup karar veriyoruz. Daha sonra değiştirilmesi gereken birşey varsa defect açıyoruz.
		K3_UY_1	Kullanılabilirlik sonuçlarını her aşamada <u>yazılım teknik liderlerine</u> geri bildirim sağlamada kullanıyoruz.
		K6_TM_2	Kullanılabilirlik değerlendirme sonuçlarını <u>product ekibine</u> analiz ve test aşamasında, <u>R&D'ye development ve test sırasında.</u>
KG_3_1	Kullanılabilirlik değerlendirmesinin yazılım geliştirme sürecinde uygulanm asının avantajları ve dezavantajlar	K1_YG_1	Kullanılabilirliği mümkün olmayan yazılımlar için <u>zaman ve para kaybı olmasını</u> önlemesi açısından, kullanılabilirlik değerlendirmesi yazılım geliştirme sürecinde uygulanmalı.
		K1_YG_2	<u>Müşteri memnuniyeti</u> sağlamaı, ürünün daha çok satma potansiyeli oluşturmaı. <u>Zaman alıcı</u> bir çalışma olması.
		K4_KU_1	<u>Kullanıcıya daha yakın önyüzler</u> tasarlanıyor. Ama aldığı süreyi iyi değerlendirmek proje takvimine uymak gerekiyor.
		K3_İA_1	Kullanılabilirlik süreçleri projeye dahil edilmediğinde <u>zaman maliyetini</u> düşürmek mümkün ancak bu şekilde ürünün <u>kullanıcılar tarafından etkin bir şekilde kullanılma ihtimali</u> düşüyor.
		K2_YG_3	Dezavantajı zaman kaybı olabilir. Avantajı <u>ise kullanıcı memnuniyeti</u> sağlanıyor.
		K5_TM_1	
		K3_UY_1	<u>Dezavantajı olacağını düşünmüyorum.</u> Yazılım sürecinde uygulanması <u>müşterilerinin beklentilerini daha iyi</u>

			<u>karsılayacağını</u> düşünüyorum.
		K6_TM_2	Avantajları: <u>Kolay anlaşılır fonksiyonallite, gereksiz geliştirmeden kurtulma, daha az bug ve daha az bakım maliyeti, daha kolay otomasyon</u> , Dezavantajları: <u>Daha fazla araştırma ve hazırlanma (analiz, dokümantasyon vb.)</u> süreci gerektiriyor. Dolayısıyla <u>zaman ve yetkin personel</u> gerektiriyor.
KG_3_2	Kullanılan yazılım yaşam döngüsüne kullanılabilirlik entegrasyonu	K1_YG_1	Yazılım sürecine çok fazla <u>entegre edildiğini düşünmüyorum</u> .
		K1_YG_2	Tam olarak <u>entegre olmadığını düşünüyorum</u> . Sebebinin gerekli önemin verilmemesi ve sistemli bir tasarım süreci olmamasına bağlıyorum.
		K4_KU_1	<u>İyi</u> durumdayız.
		K3_İA_1	<u>Hiç entegre edilmiyor</u> .
		K2_YG_3	<u>Entegre olduğunu düşünmüyorum</u> . Sadece son aşamalarda testlerle.
		K5_TM_1	<u>Entegre edildiğinin düşünmüyorum</u> . Her aşamada kontrol edilmiyor çünkü. Testçiler genelde kontrol ediyor. Analiz kısmında da gerçi bakılıyor bazen.
		K3_UY_1	<u>Yeterince olmadığını</u> düşünüyorum.
		K6_TM_2	Şimdilik <u>ortalama seviyede</u> . İleride daha etkin şekilde entegre olacak.
KG_3_3	Yazılım yaşam döngüsüne kullanılabilirlik entegrasyonunda uygulanabilecek aktiviteler	K1_YG_1	
		K1_YG_2	<u>Eğitimler</u> ile yazılım geliştirmecilerin ve takım liderlerinin kullanılabilirliğin önemini kavraması sağlanabilir.
		K4_KU_1	Analiz ve test fazlarına <u>uygun kontrol noktalarını</u> ekleyerek, <u>Analiz ve yazılım ekiplerini eğiterek</u> .
		K3_İA_1	Projeye kullanılabilirlik konusunda <u>uzman personellerin dahil edilmesi</u> ile birlikte zaman kaybına yol açmadan kullanılabilirlik süreçleri entegre edilebilir.

		K2_YG_3	
		K5_TM_1	O <u>bilinc</u> olması gerekiyor yöneticilerde. <u>Eğitim şart.</u> <u>Kullanılabilirlik eğitimleri</u> verilebilir önemini kavratmak için.
		K3_UY_1	
		K6_TM_2	Son kullanıcıya yapılan <u>testlerin artırılması, personel yetkinliğinin geliştirilmesi ve stil kılavuzlarının hazırlanıp otomasyon testleri ile</u> desteklendikçe daha etkili olacaktır.



EK-E MODEL DEĞERLENDİRMESİ GÖRÜŞMELERİNE VERİLEN CEVAPLAR

Tablo E.1

Kategori no	Kategori adı	Katılımcı kodu	Cevaplar
KG_1	Kullanıcı deneyimi konusunda tecrübeler	PG1_T1	Kullanıcı en önemli faktör. Paydaşların yorumunu almak çok önemli. Son çalıştığım firma uluslararasıydı ve yabancı ortaklar vardı. Ülkelerine göre süreçleri değişebilir. Kişilerin psikolojik durumları çok önemlidir. Daha önce PMI, PMBOK, PMP projelerinde yer aldım. RACI:Görev atama matrisi kullanılmaktaydı.
		PG1_T2	Yazılım yaparken analiz gruplarının içinde yer almadım. Tecrübem yok.
		PG1_T3	Tecrübem yok. Genellikle analist ve test grubunun görevi.
		PG2_T1	Deneyimim yok.
KG_2_1	Kullanılan çevik yöntemler	PG1_T1	Çevik yöntemlerden SCRUM konusunda tecrübem var.
		PG1_T2	Ders Projesi sayesinde öğrendim.
		PG1_T3	Hayır. Tecrübem yoktu.
		PG2_T1	Evet tecrübem vardı. Scrum kullandım. Sprint Meeting, Planning, Daily Review Meeting, Sprint Review Meeting.
KG_2_2	Çevik yöntemlerden TDD/BDD hakkında tecrübe	PG1_T1	İlk defa ders projesinde kullandım. Müşterinin istediğiyle yazılımcıların görebildikleri birbirinden farklıdır. Müşteriyi işin içine girer girmez çekmek zorundasınız. DOG(BDD) buna göre çok yazılımsal kalabilir. Kullanıcı DOG(BDD) la direk muhatap olursa, anlayamadığı bir şeyi yorumlayamayabilir.
		PG1_T2	İlk ders projesinde kullandım.
		PG1_T3	Ders projesinde.
		PG2_T1	Biraz bilgim var. Gerçek projelerde hiç kullanmadım.
KG_3_1	Ders projesinde gerçekleştirilen süreç ve takım çalışması	PG1_T1	İlk önce farklı personalardan gereksinimleri topladık. Toplanan gereksinimleri KOD senaryolarına dönüştürdük. Senaryolara göre ekran tasarımlarını yapıp kullanıcı testlerini gerçekleştirdik.
		PG1_T2	Farklı kullanıcı profilleri oluşturduk. Gereksinimleri topladık. Gereksinimleri KOD senaryolarına dönüştürdük. Ekranları tasarımlarını ve testlerini yaptık. İnsan kaynakları projesi daha görsel iken , Finansal bir proje olduğunda daha görsellikten uzak. Kafada

			canlandırılabilmesi, soyutlaşması adına önce BDD sonra prototipleme olması yerine bazı durumlarda protoriplemeyi öne almayı öneriyorum.
		PG1_T3	Önerilen <u>modelin analiz ve tasarım safhasını</u> gerçekleştirdik. <u>Gereksinimlerin</u> toplanıp <u>KOD</u> senaryolarına dönüştürülmesi ve <u>ekran tasarımlarını</u> yaratıp <u>kullanıcı testlerini</u> yaptık.
		PG2_T1	Önce gereksinimleri topladık ve kullanıcı personaların oluşturduk. Sonra <u>kullanıcı senaryolarını vazdık cucumber kullanarak</u> . Ve kullanıcı senaryosuna göre <u>prototip</u> oluşturduk. <u>Sonunda prototip üzerinden kullanıcı testi yaptık.</u> http://ing.cloudcontrolled.com/ burden ulaşabilirsin.
KG_3_2	DOG(BDD)'nin sürece katkısı	PG1_T1	<u>Kolay bir dildi.</u> Çıkarılan personaların <u>KOD</u> senaryosu yazımına katkısı oldu. Personalalar oluşturulurken caseler ele alınıyor. <u>Senaryolar yazılırken de tüm caseler düşünülmüş oluyor.</u>
		PG1_T2	Olayların sırası <u>KOD senaryosu</u> ile önceden düşünüldüğünden <u>prototiplemeler</u> (ekranlar) daha cabuk oluşturuldu.
		PG1_T3	<u>KOD</u> senaryoları aslında projenin <u>canlı dökümantasyonunu</u> da sağlayan bir yaklaşım diyebiliriz. Geliştirilmek istenen, sistemde yer alacak her gereksinim ayrıntılı şekilde <u>KOD</u> senaryolarıyla dökümante edilmektedir.
		PG2_T1	Yazılan KOD senaryolarıyla <u>daha kolay prototipleme</u> yapılabildi.
KG_4	DOG(BDD)'nin kullanıcı odaklı tasarıma katkısı	PG1_T1	Paydaşlar <u>arasında ortak dille konuşulmasını sağlıyor.</u>
		PG1_T2	Her senaryo ayrıntılı olarak düşünüldüğünden ve senaryolara <u>döküldüğünden yanlış yapma olasılığı azalıyor ve kullanıcının hangi adımda ne yapacağı önceden</u> kestirilebiliyor.
		PG1_T3	<u>KOD</u> senaryolarını oluştururken <u>kullanıcı hikayeleri</u> üzerinde durduk. Aslında her <u>kullanıcı hikayesi senaroların özet bilgisiydi</u> ve kullanıcı bilgisini de veriyordu. <u>Böylece roller iyice kavranmış</u> oldu. Roller daha iyi kafada canlandığından kendimizi onun yerine koyup <u>senaryoları çoğaltabildik.</u> Ve KOD senaryoları kullanılabilirlik testinde kullandığımız senaryolar oldu.
		PG2_T1	Behavior Driven Development kendi adından da belli olduğu gibi <u>kullanıcının sistemde neyin nasıl yapmak istediğini belirttiği</u> için. Senaryo ve featurelar da ona göre belli olur.

KİŞİSEL YAYIN VE ESERLER

- [1] Durdu P. O., **Coskan N. D.**, Kullanılabilirlik ve Yazılım Yaşam Döngüsü: Türkiye'deki Yazılım Organizasyonlarındaki Durum, 8. *Ulusal Yazılım Mühendisliği Sempozyumu*, UYMS, ODTÜ, KKTC, 8-10 Eylül 2014.



ÖZGEÇMİŞ

1989 yılında Kocaeli’de doğdu. İlk, orta ve lise öğrenimini Kocaeli’de tamamladı. 2008 yılında girdiği Kocaeli Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü’nden 2012 yılında Bilgisayar Mühendisi olarak mezun oldu. 2011 yılında Teknopark bünyesinde hizmet veren P.I.Works isimli şirkette işe girmiş olup, burada çalışmaya halen devam etmektedir. 2013 yılında başladığı Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı’nda Yüksek Lisans eğitimine devam etmektedir.

