UYGULAMALARIN MOBİL VE MASAÜSTÜ SÜRÜMLERİNİN KOD TABANLI KARŞILAŞTIRMASI: KEŞİFSEL BİR ÇALIŞMA

CODE-BASED COMPARISON OF SOFTWARE APPLICATIONS' MOBILE AND DESKTOP VERSIONS: AN EXPLORATORY STUDY

SENA SÖNMEZ ÇİÇEK

ÖĞR. ÜYESİ DR. AYÇA TARHAN
Tez Danışmanı

Hacettepe Üniversitesi
Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin
Bilgisayar Mühendisliği Anabilim Dalı için Öngördüğü
YÜKSEK LİSANS TEZİ olarak hazırlanmıştır.

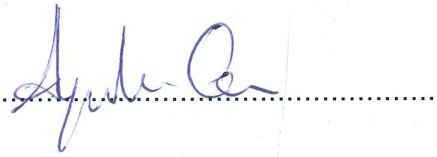
2018

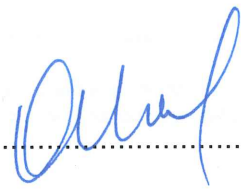
Sena SÖNMEZ ÇİÇEK tarafından hazırlanan **"Uygulamaların Mobil ve Masaüstü Sürümlerinin Kod Tabanlı Karşılaştırması: Keşifsel Bir Çalışma"** adlı bu tez çalışması aşağıdaki jüri tarafından **BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI** 'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Doç. Dr. Altan KOÇYİĞİT
Başkan ..... 

Dr. Öğretim Üyesi Ayça TARHAN
Danışman ..... 

Doç. Dr. Oumout CHOUSEINOGLOU
Üye ..... 

Doç. Dr. Aysu BETİN CAN
Üye ..... 

Dr. Öğretim Üyesi Fuat AKAL
Üye ..... 

Bu tez Hacettepe Üniversitesi Fen Bilimleri Enstitüsü tarafından **YÜKSEK LİSANS TEZİ** olarak onaylanmıştır.

Prof. Dr. Menemşe GÜMÜŞDERELİOĞLU
Fen Bilimleri Enstitüsü Müdürü

YAYIMLAMA VE FİKRİ MÜLKİYET HAKLARI BEYANI

Enstitü tarafından onaylanan lisansüstü tezimin/raporumun tamamını veya herhangi bir kısmını, basılı (kağıt) ve elektronik formatta arşivleme ve aşağıda verilen koşullarla kullanıma açma iznini Hacettepe Üniversitesine verdiğimi bildiririm. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanım hakları bana ait olacaktır.

Tezin kendi orijinal çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanılması zorunlu metinlerin yazılı izin alınarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

Yükseköğretim Kurulu tarafından yayınlanan ***"Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge"*** kapsamında tezim aşağıda belirtilen koşullar haricince YÖK Ulusal Tez Merkezi / H.Ü. Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

- Enstitü / Fakülte yönetim kurulu kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren 2 yıl ertelenmiştir. [1]
- Enstitü / Fakülte yönetim kurulunun gerekçeli kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren ... ay ertelenmiştir. [2]
- Tezimle ilgili gizlilik kararı verilmiştir. [3]

03./..09.../.2018

(İmza)

Öğrencinin Adı SOYADI

Sena Sönmez Gigtik

[i]

---

[i] *"Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge"*

*(1) Madde 6. 1. Lisansüstü tezle ilgili patent başvurusu yapılması veya patent alma sürecinin devam etmesi durumunda, tez **danışmanının** önerisi ve **enstitü anabilim dalının** uygun görüşü üzerine **enstitü** veya **fakülte yönetim kurulu** iki yıl süre ile tezin erişime açılmasının ertelenmesine karar verebilir.*

*(2) Madde 6. 2. Yeni teknik, materyal ve metotların kullanıldığı, henüz makaleye dönüşmemiş veya patent gibi yöntemlerle korunmamış ve internetten paylaşılması durumunda 3. şahıslara veya kurumlara haksız kazanç imkanı oluşturabilecek bilgi ve bulguları içeren tezler hakkında tez **danışmanının** önerisi ve **enstitü anabilim dalının** uygun görüşü üzerine **enstitü** veya **fakülte yönetim kurulunun** gerekçeli kararı ile altı ayı aşmamak üzere tezin erişime açılması engellenebilir.*

*(3) Madde 7. 1. Ulusal çıkarları veya güvenliği ilgilendiren, emniyet, istihbarat, savunma ve güvenlik, sağlık vb. konulara ilişkin lisansüstü tezlerle ilgili gizlilik kararı, **tezin yapıldığı kurum** tarafından verilir *. Kurum ve kuruluşlarla yapılan işbirliği protokolü çerçevesinde hazırlanan lisansüstü tezlere ilişkin gizlilik kararı ise, **ilgili kurum ve kuruluşun önerisi** ile **enstitü** veya **fakültenin** uygun görüşü üzerine **üniversite yönetim kurulu** tarafından verilir. Gizlilik kararı verilen tezler Yükseköğretim Kuruluna bildirilir.*
Madde 7.2. Gizlilik kararı verilen tezler gizlilik süresince enstitü veya fakülte tarafından gizlilik kuralları çerçevesinde muhafaza edilir, gizlilik kararının kaldırılması halinde Tez Otomasyon Sistemine yüklenir

** Tez **danışmanının** önerisi ve **enstitü anabilim dalının** uygun görüşü üzerine **enstitü** veya **fakülte yönetim kurulu tarafından karar verilir.***

ETİK

Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırlamış olduğum bu tez çalışmasında,

- tez içerisinde kullanmış olduğum bütün bilgileri ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- yazılı ve görsel olarak sunduğum bütün bilgi ve sonuçların bilimsel ahlak kurallarına uygun olduğunu,
- tez çalışmam sırasında çalışmalarından yararlandığım kişilerin çalışmalarına bilimsel normlara uygun bir şekilde atıfta bulunduğumu ve bu çalışmaları tez çalışmam içerisinde kaynak olarak gösterdiğimi,
- elde ettiğim sonuçlar üzerinde hiçbir şekilde tahrifat yapmadığımı ve
- tez çalışmamın hiçbir bölümünü başka bir üniversitede ve bu üniversitede ayrı bir tez çalışması olarak sunmadığımı

beyan ederim.

03/09/2018

SENA SÖNMEZ ÇİÇEK

ÖZET

UYGULAMALARIN MOBİL VE MASAÜSTÜ SÜRÜMLERİNİN KOD TABANLI KARŞILAŞTIRMASI: KEŞİFSEL BİR ÇALIŞMA

Sena SÖNMEZ ÇİÇEK
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Danışmanı: Öğr. Üyesi Dr. Ayça TARHAN
Tez Eş Danışmanı: Doç. Dr. Vahid GAROUSİ
Eylül, 2018, 67 sayfa

Belirli bir yazılım platformunun özellikleri, genellikle, bu platform için geliştirilen uygulamaların tasarımını ve kaynak kodunu etkilemektedir. Aynı uygulamayı hem masaüstü hem de mobil platform için üretecek geliştiriciler için platformun uygulamalar üzerinde nasıl bir etkisi olduğu bilmek faydalı olabilir. Literatürde Android, iOS ve BlackBerry gibi mobil platformların ve bu platformlar için geliştirilmiş uygulamaların farklı açılardan karşılaştırıldığı birçok araştırma bulunurken bir uygulamanın, mobil sürümlerine karşı masaüstü sürümlerinin özelliklerini değerlendiren veya karşılaştıran araştırmalar daha az sayıdadır. Bu tez ile mobil ve masaüstü platformlar için geliştirilmiş uygulamalar; tasarım ölçütleri, kaynak kod boyutu ve platform bağımlılığı açısından incelenmiş ve bu platformlar arasında bir benzerlik kurulup kurulamayacağı araştırılmıştır. Bu doğrultuda keşifsel araştırma yöntemini benimseyerek dört araştırma sorusu oluşturulmuştur: (1) Mobil ve masaüstü uygulamaların tasarımı, nesneye yönelik ölçütler bakımından nasıldır? (2) Mobil ve masaüstü uygulamaların büyüklüğü, kaynak kod boyutu bakımından nasıldır? (3) Mobil ve masaüstü platformlar için geliştirilmiş uygulamalar, farklı türden bağımlılıklar açısından nasıldır? (4) Bu iki platform için geliştirilmiş uygulamalar arasında benzerlik gözetilebilir mi? Bu soruları cevaplamak için, açık kaynaklı iki uygulamanın mobil ve masaüstü

sürümlerinin kaynak kodları incelenmiş ve karşılaştırmalı olarak analiz edilmiştir. Tasarım değerlendirmesinde yararlanılan C&K ölçüt kümesinde yer alan, DIT ve WMC ölçütleri için elde edilen sonuçların masaüstü sürümlerde daha yüksek olduğu, RFC ve CBO ölçütü için elde edilen sonuçların ise mobil sürümlerde daha yüksek olduğu saptanmıştır. İkinci araştırma sorusu kapsamında her iki uygulama çifti için mobil sürümlerin kod boyutunun daha büyük olduğu gözlenmiştir. Üçüncü araştırma sorusu ile mobil sürümlerin üzerinde çalıştığı platforma daha bağımlı olduğu sonucu elde edilmiştir. Son araştırma sorusu ile diğer araştırma sorularının cevapları aranırken elde edilen veriler ve araştırmanın yapıldığı koşullar değerlendirildiğinde ise bu çalışmada incelenen, farklı dillerle geliştirilmiş denk özelliklere sahip mobil ve masaüstü uygulama çiftlerinin, platform benzerliğini tartışarak çıkarımlar yapmak için elverişli olmadığı sonucuna varılmıştır.

Anahtar Kelimeler: Yazılım platformu, platform etkisi, mobil yazılım, masaüstü yazılım, yazılım tasarımı, yazılım ölçütleri, keşifsel çalışma

ABSTRACT

CODE-BASED COMPARISON OF SOFTWARE APPLICATIONS' MOBILE AND DESKTOP VERSIONS: AN EXPLORATORY STUDY

Sena SÖNMEZ ÇİÇEK
Master Degree, Department of Computer Science
Supervisor: Asst. Prof. Dr. Ayça TARHAN
Co-supervisor: Assoc. Prof. Dr. Vahid GAROUSI
September, 2018, 67 pages

Characteristics of a given software platform usually have profound effects on the design and source code of applications developed in that platform. There are fewer studies evaluating or comparing the features of desktop versions against mobile versions of an application while there are many studies in the literature comparing mobile platforms such as Android, iOS and BlackBerry, and applications developed for these platforms from different angles. With this thesis, applications developed for mobile and desktop platforms have been examined in terms of design criteria, source code size and platform dependency and it has been researched whether a similarity can be established between these platforms. In this direction, four research questions were constructed by adopting the method of exploratory research: (1) How is the design of mobile and desktop applications, in terms of object oriented metrics? (2) How is the size of mobile and desktop applications in terms of source code size? (3) How are applications developed for mobile and desktop platforms different in terms of dependencies? (4) Can similarities be identified between the applications developed for these two platforms? To answer these questions, the source code for the mobile and desktop versions of two open source applications has been examined and

analyzed in a comparative way. It was found that the results obtained for the DIT and WMC metrics in the C & K metric set, which is used in the design evaluation, are higher in the desktop applications, and the values for the RFC and CBO metrics are higher in the mobile applications. Within the scope of the second research question, it was observed that the code size of mobile versions was larger than desktop versions for both application pairs. With the third research question, it is concluded that the mobile applications are more dependent on the platform. When the results of the last research question and the other research questions are evaluated, it is concluded that the obtained data and the conditions of the research are not suitable for making inferences by discussing the similarity of the platforms with feature-equivalent mobile and desktop application couples which have been developed with different languages.

Keywords: Software platform, platform effect, mobile software, desktop software, software design, software metrics, exploratory study

TEŞEKKÜR

Tez çalışmam boyunca değerli bilgi birikimi ve tecrübesi ile bana yol gösteren, güleryüzü ve pozitif enerjisi ile desteğini hiçbir zaman esirgemeyen çok değerli danışmanım Sayın Dr. Öğretim Üyesi Ayça TARHAN'a çok teşekkür ederim. Tez savunma sınavı sırasında önemli ve değerli yorumlarıyla katkıda bulunan sayın jüri üyelerim Doç. Dr. Altan KOÇYİĞİT, Doç. Dr. Aysu Betin CAN, Doç Dr. Oumout CHOUSEINOGLOU, Dr. Öğretim Üyesi Fuat AKAL ve Doç. Dr. Murat ERTEN'e saygılarımı ve teşekkürlerimi sunarım.

Tez çalışmam boyunca beni motive eden ve yardımlarını esirgemeyen arkadaşım Burcu YALÇINER'e çok teşekkür ederim.

Eğitim hayatım boyunca hep yanımda olan, maddi ve manevi desteklerini asla esirgemeyen canım aileme ve hayatımın son 6 yılında hep yanımda olan ve bu zorlu tez sürecinde bana inanıp, beni hiç yalnız bırakmayan sevgili eşime sonsuz teşekkür ederim. Son olarak, henüz kavuşmadığım, sağlıkla dünyaya gelmesini dilediğim oğluma, annesi için bu süreci zorlaştırmadığı ve pes etmeden yoluma devam etmemi sağladığı için teşekkür ederim.

İÇİNDEKİLER

Sayfa

ÇİZELGELER

ŞEKİLLER

SİMGELER VE KISALTMALAR

Kısaltmalar

API	Application Programming Interface
CBO	Coupling Between Objects
DIT	Depth of Inheritance Tree
GPS	Global Positioning System
GSM	Global System for Mobile Communications
KSS	Kod Satır Sayısı
LCOM	Lack of cohesion in methods
LOC	Line of Code
NOC	Number of Children
PBO	Platform Bağımlılık Oranı
PC	Personal Computer
PDR	Platform Dependency Ratio
RFC	Response for a Class
WMC	Weighted Methods per Class

1. GİRİŞ

Mobil cihazların, sağladıkları kullanım kolaylığı, bilgiye hızlı ulaşma imkânı ve kullanıcı talepleri doğrultusunda artan uygulama sayıları ile yaşantımızdaki yeri her geçen gün daha da büyümektedir. 2016 yılında tüm dünyada indirilen mobil uygulama sayısı 149,3 milyar olarak raporlanmıştır; 2021 yılında ise bu sayının 351 milyar olacağı tahmin edilmektedir. Uygulama erişim platformu olan Google Play'de 2013'te 1 milyon olan uygulama sayısı ise 2017 Aralık'ta 3,5 milyona ulaşmıştır [1]. Öte yandan masaüstü ve dizüstü bilgisayarlar da artan performans kapasiteleri ve özellikle iş ve eğitim hayatında duyulan ihtiyaç nedeniyle hayatımızdaki yerini korumaktadır. Uygulama geliştiriciler açısından ise bu durum, piyasa ve kullanıcı beklentilerini karşılamak için uygulamaları hem mobil hem de masaüstü platformlar için geliştirmeyi ihtiyaç haline getirmiştir. Bilgisayar işletim sistemlerinin aksine, mobil cihazlar için geliştirilmiş işletim sistemleri donanımları, depolama alanları, güç dağılımı ve mobilite koşulları ile sınırlandırılmıştır. Bilgisayar uygulamalarının geliştirilmesiyle karşılaştırıldığında, mobil cihaz işletim sistemlerinde, uygulamaların geliştirme sırasında dikkate alınması gereken farklı özellikleri (bellek, ekran boyutu vb.) vardır [2].

Aynı uygulamayı her iki platform için de geliştirmek gerektirdiği kaynak, zaman ve çaba açısından zorlu bir iştir. Her platformun desteklediği kendine özgü bir geliştirme ortamı, geliştirme dili ve geliştirme teknolojisinin olması, bir uygulamayı birden fazla platform için geliştirirken önem kazanmaktadır. Bunun yanı sıra, özellikle mobil ve masaüstü uygulamaları göz önünde bulundurduğumuzda cihazların bellek kapasitesi, ekran boyutu, işlemci hızı gibi farklı özelliklerinin olması da ilgili platform ve cihaz türü için uygulama geliştirirken belirleyici faktörlerdir. Tüm bunlar değerlendirildiğinde, belirli bir yazılım platformuna ait özelliklerin, bu platformda geliştirilen uygulamalar üzerinde etkiye sahip olduğu açıktır. Bu etkiler, yazılımın mimarisi, tasarımı ve kaynak kodu üzerinde gözlemlenebilir [3]. Aynı uygulamayı hem masaüstü hem de mobil platformlar için geliştirirken yapılan gözlemler, geliştiricilere kendi yazılım kararlarını almada bakış açısı sağlayabilir.

Literatürde, iOS, Android ve BlackBerry platformların birbiri arasında karşılaştırılıdığı birçok çalışma yer almaktadır. Örneğin, Syer ve arkadaşları [4],

Android ve BlackBerry uygulamaları kod tabanlı bir değerlendirme yaparak karşılaştırmıştır. Mobil platformlar için uygulama geliştirirken kısıtlı kaynaklarla birden fazla platformu hedefleyen geliştiriciler için, üç mobil uygulama çiftini kaynak kodu boyutu, bağımlılık ve kod değişiklik (İng. code churn) ölçütlerini üzerinden incelemişlerdir. Vaka çalışmalarında, Android platformu için geliştirilmiş uygulamalarının üzerinde çalıştığı platforma daha fazla bağımlı olduğunu ve BlackBerry platformu için geliştirilmiş uygulamalara kıyasla, daha küçük bir kod boyutuna sahip olduğunu bulmuşlardır. Syer ve arkadaşları bir diğer çalışmalarında [5], açık kaynak kodlu 15 popüler mobil uygulama ile 5 masaüstü / sunucu uygulamasını (2 büyük uygulama, 3 küçük Unix Yardımcı Programı) kaynak kod boyutu ve hata giderme süresi açısından incelemiştir. Mobil uygulama geliştiricilerin, hataları masaüstü / sunucu uygulama geliştiricilerine kıyasla daha kısa sürede düzeltme eğiliminde olduklarını raporlamışlardır. Başka bir çalışmada, Syer ve arkadaşları mobil uygulamalar üzerinde, platform bağımlılığı ve hataya yatkınlık ilişkisini incelemişlerdir [6]. Hataya yatkın dosyaların platforma daha bağımlı olduğunu ve platform bağımlılığının hataya yatkınlığı artırdığını raporlamışlardır.

Literatürde mobil uygulamalar ve masaüstü uygulamaları bir arada değerlendiren oldukça az sayıda çalışmaya rastlanmıştır. Bu çalışmalar genellikle kullanıcılara yöneliktir ve uygulama geliştiricileri hedef alan çalışmalar, daha da az sayıdadır. Syer ve arkadaşlarının gerçekleştirdiği çalışmada [5], mobil ve masaüstü uygulamalar karşılaştırılmış fakat uygulama denkliği gözetilmemiştir. Çalışmalar çoğunlukla mobil platformlara (iOS, Android, BlackBerry) ve bu platformların kendi içinde değerlendirmesine odaklanmıştır. Bu nedenle mobil ve masaüstü uygulamaların bir arada değerlendirilmesi konusunda literatürdeki birikimin yetersiz olduğu söylenebilir.

Keşifsel araştırma yöntemi, varsayımsal veya kuramsal bir fikri araştırırken başlangıç adımı olarak tanımlanır. Amaç yeni bilgileri keşfetmek ve yaymaktır. Bir konuda belirsiz bir problem olduğunda, o alanda var olan sonuçlar olgunlaşmamış veya kesin değilse ve araştırma süreci tanımlanmamış ya da erken aşamalarda ise keşifsel araştırma yöntemi önerilir [7]. Oivo ve arkadaşları [8], Govender'a göre keşifsel araştırmanın, bir olgu hakkında veri toplamak ve mevcut çalışmaları tekrar

etmek yerine derinlemesine bir bakış sağlamayı hedeflediğini söyler. Amaç, gelecek çalışmalar için öncelikleri belirlemeye yardımcı olmak ve var olan olgu üzerinde yeni hipotezler ortaya koymaktır.

Mobil ve masaüstü uygulamaların geliştirici bakış açısıyla ele alındığı çalışmaların azlığı ve hipotez yetersizliği nedeniyle, bu alanda yapılacak keşifsel bir çalışma ile geliştiricilere ve gelecekte bu alanda çalışmak isteyen araştırmacılara faydalı olunabilir. Bu tez çalışmasında, mobil sürümleri Java ile geliştirilmiş, masaüstü sürümleri ise C++ ve QT ile geliştirilmiş iki uygulama çifti; nesneye yönelik tasarımı, kaynak kod boyutu ve platform bağımlılığı bakımından incelenmiştir. Bulgulara dayanarak mobil ve masaüstü platform için geliştirilmiş uygulamalar arasında benzerlik gözetilip gözetilemeyeceği araştırılmıştır. Keşifsel yöntemle elde edilecek sonuçların, mobil ve masaüstü platformlar için benzer teknolojiler ile uygulama geliştirecek olanlara fikir vermesi ve gelecekte bu alanda çalışmalar yapacak araştırmacılara bir bakış açısı sağlayarak yeni araştırma alanları belirlemeye yaraması beklenebilir.

Bu tez çalışması kapsamında, dört araştırma sorusu belirlenmiştir. Araştırma soruları şu şekildedir;

- (AS1) Mobil ve masaüstü uygulamaların tasarımı, nesneye yönelik ölçütler bakımından nasıldır?
- (AS2) Mobil ve masaüstü uygulamaların büyüklüğü, kaynak kod boyutu bakımından nasıldır?
- (AS3) Mobil ve masaüstü platformlar için geliştirilmiş uygulamalar, farklı türden bağımlılıklar açısından nasıldır?
- (AS4) Bu iki platform için geliştirilmiş uygulamalar arasında benzerlik gözetilebilir mi?

Bu tez çalışmasının geri kalan kısmı şu şekilde organize edilmiştir: Bölüm 2, bu alandaki ön bilgiyi ve ilişkili çalışmaları sunmaktadır. Bölüm 3, araştırma tasarımını anlatmakta ve Bölüm 4’te çalışmamızın gerçekleştirimi ve bulguları verilmektedir. Bölüm 5’te bulguların tartışması ve geçerlilik tehditleri ele alınmaktadır. Son olarak Bölüm 6’da, çalışmamızın sonuçları ve gelecek çalışmalar için önerilerimiz özetlenmiştir.

2. ÖN BİLGİ

Bu bölümde, alanda yapılan çalışmalar için altyapı oluşturabilecek bilgiler; keşifsel araştırma yöntemi, nesneye yönelik tasarım ve nesneye yönelik tasarım ölçütleri, bağımlılık ve platform bağımlılığı, Android uygulama çerçevesi ve masaüstü uygulama geliştirme alt başlıkları altında verilmektedir.

2.1 Keşifsel Araştırma Yöntemi

Belirli bir araştırma projesi için araştırma yönteminin seçimi mevcut kaynaklar, konulara erişim, ilgili değişkenleri kontrol etme fırsatı ve araştırmacıların becerileri dâhil olmak üzere birçok yerel duruma bağlıdır. Uygun bir araştırma yöntemini seçmenin ilk adımlarından biri, araştırma amacını ve sorularını netleştirmektir. Bir araştırma çalışmasının ilk aşamalarında, genellikle olay (İng. phenomena) anlaşılmaya çalışıldığından ve anlayışı açıklığa kavuşturacak yararlı farklar belirlendiğinden, keşif soruları sorulması gerekir [7]. Keşif amaçlı soruları yanıtlamaya yönelik araştırma yöntemleri, tematik kuramlar oluşturulmasına yardımcı olan zengin, niteliksel veriler sunanlar olma eğilimindedir. Keşifsel araştırmaya yönelik soru kalıpları üç ana kategoride tanımlanabilir:

- Varlık soruları: “X var mıdır?”
- Tanımlama ve sınıflandırma soruları: “X nasıldır?”,”X’in özellikleri nelerdir?”, “Nasıl kategorize edilebilir?”, “Nasıl ölçülebilir” vb.
- Tanımlayıcı-karşılaştırıcı sorular: “X, Y’den nasıl farklılaşır?”. Bu tipteki sorular, iki veya daha fazla olay arasındaki benzerlikleri ve farklılıkları araştırır.

Bu soruların cevapları; kuramsal terimlerin daha kesin tanımları, bunların ölçülebildiğine dair kanıtlar ve tedbirlerin geçerli olduğuna dair kanıtlar da dahil olmak üzere, incelenen olayı daha açık bir şekilde anlatabilmektedir.

Yazılım mühendisliği literatüründe, vaka çalışmasının tanımına dair bir karışıklık söz konusudur. Vaka çalışması terim olarak genellikle uygulamalı/işlenmiş bir örnek anlamına gelir. Deneysel (İng. empirical) bir yöntem olarak, bir vaka çalışması oldukça farklıdır. Yin (2002), vaka çalışmasını “gerçek olay bağlamında, özellikle olay ve bağlam arasındaki sınırlar açıkça belli olmadığında, çağdaş bir olguyu araştıran deneysel bir araştırma” olarak sunmuştur. Vaka çalışmaları, belirli

olayların nasıl ve neden olduğu hakkında derinlemesine bir anlayış sunmaktadır ve neden-sonuç ilişkilerinin meydana geldiği mekanizmaları ortaya çıkarabilir. Vaka çalışmaları keşifsel ve doğrulayıcı olarak iki şekilde yapılabilir. Keşifsel vaka çalışmaları, yeni hipotezler türetmek ve kuramlar oluşturmak için bazı olayların ilk incelemeleri olarak kullanılmış ve doğrulayıcı vaka çalışmaları ise, var olan teorileri test etmek için kullanılmıştır.

2.2 Nesneye Yönelik Tasarım

Nesneye yönelik mimari, bir uygulama veya sistem için sorumlulukların, yeniden kullanılabilir ve kendi kendine yeterli nesneler olarak bölünmesine dayanan bir tasarım yaklaşımıdır. Nesneler, kendileriyle ilgili veri ve davranışları içerir. Nesneye yönelik tasarım, sistemi bir dizi rutin veya prosedür talimatı yerine, işbirliği ve etkileşim halindeki bir nesne kümesi olarak görür. Nesneler ayrık, bağımsız ve gevşek bağlanmış olup arabirimler aracılığıyla, metotları kullanarak veya diğer nesnelerdeki özelliklere ulaşarak ve ileti gönderip alarak iletişim kurarlar. Nesneye yönelik mimarinin 6 temel ilkesi bulunur. Bunlar şöyledir:

- *Soyutlama* (İng. abstraction): Karmaşık bir operasyonu, operasyonun temel özelliklerini koruyarak genelleştirmek olarak tanımlanır. Örneğin, soyut bir arayüz tanımlamak, Get ve Update gibi basit yöntemler kullanarak veri erişim işlemlerini destekleyen bir soyutlama biçimidir. Başka bir soyutlama biçimi, yapısal verileri barındıran iki format arasında bir eşleme sağlamak için üst veri kullanmak olabilir.
- *Kompozisyon* (İng. composition): Nesneler diğer nesnelerden oluşabilir ve bu iç nesneleri diğer sınıflardan gizlemeyi veya basit arayüzler olarak göstermeyi seçebilir. Bu ilke, kompozisyon olarak tanımlanır.
- *Kalıtım* (İng. inheritance): Nesneler diğer nesnelerden kalıtım alabilir ve temel nesnedeki işlevsellik kullanılabilir veya bu işlevler yeniden tanımlanarak özelleştirilebilir. Kalıtım ile temel nesneye yapılan değişiklikler, kalıtımı alan nesnelere otomatik olarak uygulandığı için kalıtım, bakım ve güncellemeleri kolaylaştırır.
- *Kapsülleme* (İng. encapsulation): Nesneler işlevselliği yalnızca yöntemler, özellikler ve olaylar aracılığıyla ortaya çıkarır ve diğer nesnelerdeki durum ve değişkenler gibi iç ayrıntıları gizler. Bu, diğer nesneler ve kodları

etkilemeden, arayüzleri uyumlu olduğu sürece nesneleri güncellemeyi veya değiştirmeyi kolaylaştırır.

- *Çok-biçimlilik* (İng. polymorphism): Bir değişken, işlev veya nesnenin birden çok biçim üstlenme yeteneğini ifade eder. Mevcut nesne ile bu nesnenin değişebilen yeni biçimlerini nesne farklılıklarına göre kullanarak uygulamadaki işlemler desteklenir.
- *Bağlaşımı koparma* (İng. decoupling): Nesnelerin uyguladığı ve kullanıcının anlayabileceği soyut bir arayüz tanımlanarak nesneler, o nesneyi kullanan sınıf/metot/arayüzden ayrıştırılabilir. Bu, arayüzün kullanıcılarını etkilemeden alternatif uygulamalar sağlamanıza olanak tanır.

2.3 Nesneye Yönelik Tasarım Ölçütleri

Nesneye yönelik tasarımının yukarıda bahsedilen çok-biçimlilik, kalıtım, soyutlama gibi temel ilkelerini ölçmeyi hedefleyen, literatürde önerilmiş farklı ölçüt kümeleri bulunmaktadır. Nesneye yönelik ölçütler ile kaynak kodu analiz ederek yazılımın tasarımı hakkında fikir sağlanır. Chidamber&Kemerer (C&K) ölçüt kümesi [9], MOOD ölçüt kümesi [10] ve QMOOD ölçüt kümesi, kabul görmüş ölçüt kümelerinden birkaçıdır [11].

2.3.1 Chidamber&Kemerer Ölçüt Kümesi

Chidamber&Kemerer ölçüt kümesi 6 ölçütten oluşur.

Sınıfın ağırlıklı metot sayısı (Weighted Methods per Class - WMC): WMC, bireysel bir sınıfın ağırlıklı olarak metot sayısını ölçer ve sınıfın karmaşıklığı ile ilişkilidir. Çok sayıda yöntem içeren sınıflar, daha fazla uygulamaya özgüdür. Bu da bir sınıfın yeniden kullanma olasılığını kısıtlar. [12].

Kalıtım ağacının derinliği (Depth of Inheritance Tree - DIT): Bir sınıfın kalıtım hiyerarşisindeki derinliği, sınıf düğümünden sınıf hiyerarşisi ağacının köküne kadar olan maksimum uzunluk olarak tanımlanır ve ata sınıflarının sayısıyla ölçülür. Çoklu kalıtım içeren durumlarda DIT, düğümden ağacın köküne kadar olan maksimum uzunluktur. Bir sınıf, hiyerarşide ne kadar derin olursa, kalıtımdan gelen yöntemlerin sayısının o kadar büyük olması beklenir ve bu, sınıfın davranışını tahmin etmeyi daha karmaşık hale getirir [9].

Alt sınıf sayısı (Number of Children - NOC): Bu ölçüt, her sınıf için, doğrudan soyundan gelen (alt sınıfların) sayısıdır. Çok sayıda çocuğun bulunduğu sınıfların değiştirilmesinin zor olduğu ve genellikle tüm çocuklarda meydana gelen değişikliklere bağlı olarak daha fazla test yapılması gerektiği düşünülmektedir.

Nesne sınıfları arasındaki bağımlılık (Coupling Between Object Classes - CBO): CBO, sınıfın bağımlı olduğu sınıfların sayısı olarak tanımlanır. Bir sınıfta tanımlanmış metotlar başka bir sınıfta tanımlı bir metot veya değişkeni kullanırsa bu iki sınıfın arasında bağımlılık olduğu söylenir. Nesne sınıfları arasındaki aşırı bağlanma, modüler tasarıma zarar verir ve yeniden kullanılabilirliği olumsuz etkiler [13]. Bir sınıf, bağımsızlığı arttıkça, başka bir uygulamada tekrar kullanılmak için daha elverişli hale gelir. Bu nedenle, sınıfların CBO değerlerinin düşük olması tercih edilir.

Sınıfın tetiklediği metot sayısı (Response For a Class - RFC): Bir sınıfın tetiklediği metot sayısı, o sınıfın bir nesnesi tarafından alınan bir mesaja yanıt olarak tetiklenebilen toplam metot sayısıdır. RFC, özellikle sınıf dışından da tetiklenebilen yöntemleri içerdiğinden aynı zamanda, sınıf ve diğer sınıflar arasındaki potansiyel iletişimin bir ölçüsüdür. Yüksek bir RFC'ye sahip sınıflar daha karmaşıktır ve bu sınıfların anlaşılması daha zordur. Test etme ve hata ayıklama da karmaşıktır.

Metotlardaki uyum eksikliği (Lack of Cohesion in Methods - LCOM): Bir sınıfın yerel yöntemlerinin, ayrık (kesişim içermeyen) kümelerinin sayısıdır. Uyum eksikliği veya düşük bağlılık sınıfın karmaşıklığını artırır ve bu da geliştirme esnasında karşılaşılabilecek hataları olasılığını artırır. Bu ölçüt, tasarımın nasıl uygulandığı ve sınıfın yeniden kullanılabilirliğiyle ilgili fikir verir.

2.4 Bağımlılık ve Platform Bağımlılığı

Bağımlılık (İng. dependency), bir yazılım parçası başka bir yazılım parçasına dayandığında kullanılan bir yazılım mühendisliği terimidir. Bir yazılım geliştirilirken, ihtiyaç duyulan işlevselliğe erişimi sağlayan, daha önceden geliştirilmiş, üçüncü parti uygulamalar dediğimiz uygulamalardan ve yazılımın üzerinde çalıştığı platforma ait olan uygulamalardan yararlanılır [14]. Uygulamalar, platform bağımlı ve platform bağımsız olarak ikiye ayrılır. Web uygulamaları platformdan bağımsız, bir diğer deyişle her platformda çalışan uygulamalara örnek olarak verilebilir. Öte

yandan masaüstü uygulamalar ve mobil uygulamalar çoğunlukla bir platform hedeflenerek geliştirilir ve platform bağımlıdırlar. Örneğin, Android işletim sistemine sahip cihazlar için geliştirilmiş bir mobil uygulama, IOS cihazlar üzerinde çalışmamaktadır. Platform bağımlılığı ne kadar fazla ise o uygulamanın başka platformlara taşınması zorlaşır ve bağımlı olduğu platformdaki değişikliklerden etkilenme olasılığı artar. Bu durum platform kilitlenmesi (İng. platform lock-in) olarak tanımlanır [4].

Syer ve arkadaşları [4], uygulamaların platform bağımlılıklarını ölçmek için Platform Bağımlılık Oranı (İng. Platform Dependency Ratio - PDR) olarak adlandırdıkları bir ölçüt önermişlerdir. Bu ölçüte göre bağımlılıklar 5 kategoriye ayrılır [4]:

- Dil bağımlılığı - dil platformunun bir parçası olan bir sınıfa bağımlılık (ör. Java.lang.Thread).
- Kullanıcı arayüzü (KA) bağımlılığı - cihaz platformunun bir parçası olan ve ayrıca kullanıcı arayüzünden sorumlu olan bir sınıfa bağımlılık.
- Platform bağımlılığı - cihaz platformunun bir parçası olan bir sınıfa bağımlılık (KA bağımlılıkları hariç).
- Üçüncü parti kütüphane bağımlılığı - üçüncü parti bir kütüphaneye ait bir sınıfa bağımlılık.
- Proje bağımlılığı – proje için geliştirilmiş bir sınıfa olan bağımlılık.

Syer ve arkadaşları “Platform Bağımlılık Oranı”nı (PDR), platform ve kullanıcı arayüzü bağımlılıklarının toplam bağımlılık sayısına oranı olarak tanımlar. Her sınıf için platform ve kullanıcı arayüzü bağımlılığı sayısının toplamı, o sınıfa ait dil, kullanıcı arayüzü, platform, üçüncü parti kütüphane ve proje sınıflarına olan bağımlılık sayısı toplamına bölünür ve tüm sınıflar için elde edilen bu oranın ortalaması alınarak uygulamanın Platform Bağımlılık Oranı (PDR) hesaplanır.

$$\mathrm{PDR} = \sum_{i=0}^{n} \left(\frac{KA_i + P_i}{D_i + KA_i + P_i + 3rd_i + PR_i} \right) / n \qquad (1)$$

KA: Kullanıcı arayüz sınıflarına bağımlılık sayısı
P: Kullanıcı arayüz sınıfları hariç platform sınıflarına bağımlılık sayısı
D: Dil sınıflarına bağımlılık sayısı
3rd: Üçüncü parti kütüphane sınıflarına bağımlılık sayısı
PR: Proje sınıflarına bağımlılık sayısı
n: Toplam sınıf sayısı

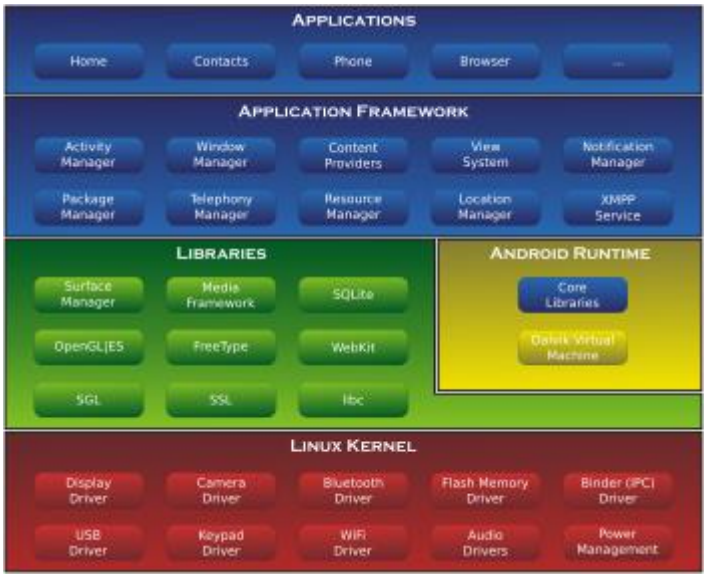
Denklem 1 ile hesaplanan platform bağımlılık oranı düşükse bu, uygulamaların platform API'larına önemli ölçüde dayanmadıklarını gösterir. Bu tür uygulamaların farklı platformlara taşınması daha kolay olabilir.

2.5 Android Uygulama Çerçevesi

Android, Google tarafından geliştirilen, Linux çekirdeğinin ve diğer açık kaynaklı yazılımın değiştirilmiş bir sürümünü esas alan ve öncelikle akıllı telefonlar ve tabletler gibi dokunmatik ekranlı mobil cihazlar için tasarlanmış bir mobil işletim sistemidir. Android 2003 yılında piyasaya çıkmıştır ve dünyanın en popüler mobil işletim sistemidir [15].

Güçlü geliştirme çerçevesi sayesinde yazılım geliştiriciler geniş cihaz yelpazesinde kendi uygulamalarını oluşturabilmektedir. Geliştiricilere yardımcı olmak için Android, Android Yazılım geliştirme kiti (SDK) sağlar. Android uygulama geliştirme için Java programlama dilini destekler. Android yazılım geliştirme kiti; bir hata ayıklayıcı, kitaplık, QEMU'ya (Quick Emulator) dayalı bir el aygıtı emülatörü (benzetici), belgeler, örnek kod ve öğreticiler içerir.

Android İşletim Sistemi, katmanlı bir sistemdir ve dört ana katmandan oluşur: çekirdek, kitaplıklar, uygulama çerçevesi ve uygulamalar [16]. Android işletim sistemi, kütüphane seviyesi ile aynı seviyede bulunan bu dört katman ve Android RunTime'dan oluşur.



Şekil 1. Android Uygulama Mimarisi [17]

Linux Çekirdeği: Android Kernel, Linux Çekirdeği'ne (Linux2.6) dayanmaktadır. En alttaki katmandır ve cihazın donanımı ile Android yığınının üst katmanları

arasında soyutlama düzeyi sağlar. Çekirdek, sanal bellek, ağ oluşturma, sürücüler ve güç yönetimi gibi temel sistem hizmetleri sağlar.

Yerel Kütüphaneler Katmanı: Linux Çekirdek katmanının üst kısmında Android'in yerli kütüphaneleri bulunur. Bu katman, cihazın farklı veri türlerini ele almasını sağlar. Veriler donanıma özgüdür. Tüm bu kütüphaneler C veya C ++ dili ile yazılmıştır. Bu kütüphanelere Java arabirimi denir

Android Runtime: Android Runtime, Dalvik Sanal makinesi ve Core Java kütüphanelerinden oluşur. Kütüphane katmanı ile aynı seviyededir. Dalvik Sanal Makinası (Dalvik Virtual Machine – Dalvik VM), Android cihazında uygulamaları çalıştırmak için kullanılan bir Java Sanal Makinesi türüdür. Dalvik Sanal Makinası, düşük işlemci gücü ve düşük bellekli ortamlar gibi mobil cihazların donanım özellikleri için Google tarafından yeniden tasarlanmış ve optimize edilmiştir.

Uygulama Çerçevesi: Uygulamalara Java sınıfları şeklinde daha üst düzey hizmetler sunar. Uygulama çerçevesi, Android uygulamalarının yeniden kullanılabilir ve değiştirilebilir bileşenlerden inşa edildiğini tanımlamaktadır [16].

Android çerçevesinde aşağıdaki temel hizmetler bulunur:

- Etkinlik Yöneticisi
- İçerik Sağlayıcıları
- Kaynak Yöneticisi
- Bildirimler Yöneticisi
- Görünüm Sistemi
- Paket Yöneticisi
- Telefon Yöneticisi
- Konum Yöneticisi

Android'in temel özelliği, bir uygulamanın başka bir uygulamaya ait bileşen öğesini (bileşene izin veriliyorsa) kullanabilmesidir. Bu gibi işlevleri yerine getirmek için Android sisteminin uygulamanın herhangi bir kısmı talep edilirken uygulamayı başlatması ve talep edilen Java nesnelerini oluşturması gerekir. Çoğu işletim sisteminin aksine, sistemin bir Android uygulamasına müdahil olabileceği tek bir nokta yoktur (örneğin, bir Android uygulamasında main() fonksiyonu yoktur).

Bunun yerine, her bileşen, sistemin bir uygulamaya girebileceği ve bileşen nesnesini bağımsız olarak oluşturabileceği farklı bir noktadır.

Dört farklı uygulama bileşeni türü vardır. Her tür farklı bir amaca hizmet eder ve bileşenin nasıl oluşturulduğu ve yok edildiğini tanımlayan farklı bir yaşam döngüsüne sahiptir. Bu bileşenler aktivite, servis, içerik sağlayıcı ve yayın alıcıdır.

Uygulama Katmanı: Uygulamalar Katmanı, Android mimarisinde en üst katmandır. Bazı uygulamalar, SMS istemci uygulaması, Çevirici, Web tarayıcısı ve Kişi yöneticisi gibi her cihazla önceden yüklenir. Bir geliştirici kendi uygulamasını yazabilir ve mevcut uygulama ile değiştirebilir. Tüm uygulamalar Java programlama dili kullanılarak yazılmıştır.

2.6 Masaüstü Uygulama Geliştirme

Masaüstü uygulamalar, web ve mobil benzerlerinden ortalama olarak daha hızlı ve daha güçlü olarak bilinir. Bir PC (Personel Computer) uygulaması geliştirirken, hedeflenen işletim sistemine bağlı olarak birçok teknoloji ve dil seçeneği vardır. Java ve C / C ++ çapraz platform çözümleri (Windows, Mac, Linux) sağlarken, C # ve .NET çerçeveleri yakın zamana kadar sadece Windows işletim sistemi için geliştirilmiş olan uygulamalar içindi.

Sadece Windows işletim sistemi hedeflendiğinde, .NET öne çıkan teknolojilerden biridir. .NET ile uygulama geliştirirken kullanıcı arayüz (User Interface) yığınları üzerinde iki seçenek vardı; .NET WPF (Windows Presentation Framework) ve .NET Windows Forms. WPF, kullanıcı arayüz karmaşıklığı, stil özelleştirme ve grafik yoğun senaryolar gerektiren Windows tabanlı masaüstü uygulamaları için tercih edilen teknolojidir. WPF ayrıca XAML görünümlerinden yararlanır. .NET Windows Forms, masaüstü uygulamaları oluşturmak için .NET Framework'teki ilk kullanıcı arayüz teknolojisidir ve birçok profesyonel masaüstü uygulaması için, hala uygun bir çözümdür. Windows Forms, basit senaryolar için WPF'den daha kolay ve daha hafiftir. Windows Forms, XAML'yi kullanmaz, bu nedenle uygulamanızı Windows Phone veya Windows Store'a genişletmek için karar vermek, kullanıcı arayüzünüzün tamamıyla yeniden yazılmasını gerektirir.

C / C++ farklı programlama dilleri olarak kabul edilmesine rağmen, C++ sadece C dilinin bir uzantısıdır ve C'ye nesneye yönelik özellikler ekler; bu nedenle C++ ve C birlikte gruplanabilirler. C++, yüksek performansı nedeniyle yazılım geliştirmek için

daha iyi bir seçenek olarak kabul edilir. Ayrıca Windows yazılımı geliştirmek için yaygın olarak kullanılmaktadır. C++, .NET gibi yüksek seviye bir yürütüm süresi (runtime) ortamına bağlı olmaksızın, hem Windows hem de Windows dışındaki çok çeşitli platformlarda birinci sınıf geliştirme deneyimi sağlar. Bu gibi yürütüm süresi (runtime) imkânı mevcut olmadığında veya belirli hedef platformlar için çok ağır olduğunda, az önce bahsedilen durum C++'ı, taşınabilir uygulamalar için tercih edilen dil haline getirmektedir.

Windows işletim sistemi için, C++ ve Win32 API kullanmak, .NET platformunun izin verdiğinden daha iyi bir kontrol sağlamaktadır. Böylece, hedef platformunun daha sıkı kontrolü sağlanarak en yüksek performans ve verimliliği elde etmek mümkün kılınmaktadır. Bununla birlikte, uygulamanızın yürütülmesiyle ilgili böyle bir kontrol düzeyinin kullanılması daha fazla özen ve dikkat gerektirir ve yürütüm süresi performansı için geliştirme verimliliğini negatif yönde etkileyebilir.

Qt, C++'a dayanan başarılı bir çapraz platform uygulama geliştirme çerçevesidir ve C++ ile masaüstü uygulama geliştirme için çoklu işletim sistemleri hedeflenirken tercih edilebilir. Qt, hem ticari hem de açık kaynaktır ve C ve C++ ile birlikte, masaüstü uygulamalarının geliştirilmesinde C++'dan daha basit bir geliştirme deneyimi sağlar.

2.7 İlişkili Çalışmalar

Mobil uygulamalar, her geçen büyüyen bir sektör olmanın yanı sıra, birçok akademik araştırma ve çalışmanın konusu olmaktadır. Kullanıcı davranışları ve eğilimler [18], farklı mobil uygulama platformları üzerine karşılaştırmalar [4], çapraz platform uygulama geliştirme çözümleri [19], statü ve eğilimler [20] çalışmaların yoğunlaştığı alanlardan birkaçıdır.

Bu tez çalışması kapsamında, öncelikli olarak literatürde yer alan uygulama geliştirme odaklı ve farklı platformları karşılaştıran çalışmalar taranmıştır. Syer ve arkadaşları, mobil uygulamaların geliştirme ve bakım süreçlerini anlamak ve sınırlı kaynaklarla farklı platformları hedefleyecek geliştiricilere yardımcı olmak amacıyla bir vaka çalışması [4] yapmıştır. BlackBerry ve Android platformları için geliştirilmiş denk özellikteki üç uygulama çiftini kaynak kodu, bağımlılık ve kod değişiklik (İng. code churn) ölçütlerinden yararlanarak karşılaştırmışlardır. BlackBerry platformu için geliştirilmiş uygulamaların BlackBerry platformuna özel API'lere daha az

dayandığını, Android platformu için geliştirilmiş uygulamaların ise Android ve Java SE API'sine daha çok dayandığını raporlamışlardır. Bu da Android uygulamalarının üzerinde çalıştığı platforma daha fazla bağımlı olduğunu göstermektedir. Bu nedenle yazarlar, bu iki platformu da hedefleyen geliştiricilerin uygulamalarını önce BlackBerry platformu için geliştirimesini, sonrasında bu uygulamayı Android platformuna taşımasını önermişlerdir. BlackBerry platformu için geliştirilmiş uygulamaların denk özellikteki Android uygulamaların iki katından fazla bir kod boyutuna sahip olduğunu bulmuşlardır. BlackBerry platformundaki ve Java 2 ME API'daki eksik işlevselliğin, BlackBerry geliştiricilerini daha fazla üçüncü parti kütüphane kullanmak durumunda bıraktığını ve bunun da kod boyutunu artırdığını ifade etmişlerdir. Yazarlar, başka bir çalışmalarında [5], 15 popüler açık kaynak mobil uygulama ile 5 masaüstü / sunucu uygulaması (2 büyük uygulama, 3 küçük Unix Yardımcı Programı) arasında bir vaka çalışması yapmış ve kaynak kod boyutu ve hata giderme sürelerini araştırmışlardır. Kaynak kod boyutunu incelerken ek olarak, uygulamaların üzerinde çalıştıkları platforma bağımlılıklarını hesaplamışlardır. Mobil uygulamaların üzerinde çalıştığı mobil platforma çok fazla bağımlı olduklarını ve mobil uygulama geliştiricilerin, hataları masaüstü / sunucu uygulama geliştiricilerine göre daha kısa sürede düzeltme eğiliminde olduklarını keşfetmişlerdir. Ayrıca, mobil uygulamaların ve Unix yardımcı programlarının, sıklıkla incelenen masaüstü / sunucu uygulamalarına kıyasla daha küçük kod tabanlarına sahip olduklarını raporlamışlardır. Başka bir çalışmalarında ise Android uygulamaları inceleyerek platform bağımlılığının yazılım kalitesi ile ilişkisini araştırmışlardır [6]. Yazarlar, hataya açık kaynak kod dosyalarının, hatasız kaynak kod dosyalarından daha fazla Android platformuna bağımlı olma eğiliminde olduğunu bulmuşlardır. Kod dosyalarının hataya yatkınlığını, platform bağımlılığının yanı sıra kod boyutu, bağlılık (İng. coupling) ve uyum (İng. cohesion) ölçütleri ile tekrar değerlendirmişlerdir. Hataya yatkınlık ile en çok ilişkili olan ölçütün platform bağımlılığı olduğunu tespit etmişlerdir. Buradan yola çıkarak, kaynak kod dosyalarında platform bağımlılığına bakılarak, hata bulma olasılığının artırılabileceğini belirtmişlerdir.

Syer ve arkadaşları, çalışmalarında denk özellikteki Android ve BlackBerry uygulamaları, kaynak kodu ve platform bağımlılığı gibi farklı boyutlardan karşılaştırmıştır [4] ve bu çalışmanın bulgularından yararlanarak başka bir

çalışmalarında [6] mobil ve masaüstü uygulamalar üzerinde, platform bağımlılığı ile birlikte kaynak kod boyutu ve hata giderme süresini araştırmışlardır. Syer ve arkadaşlarının gerçekleştirdiği bu çalışmada, mobil ve masaüstü için farklı sayılarda uygulamalar karşılaştırılmış ve uygulama denkliği gözetilmemiştir. İncelenen mobil ve masaüstü uygulamaların birbirinden farklı olması, sonuçların genellenebilirliği açısından bir tehdit oluşturmaktadır. Bu açıdan bu tez çalışmasında mobil ve masaüstü uygulamalar farklı boyutlardan karşılaştırılmış ve incelenen uygulamalar, denk özellikli uygulamalar arasından seçilmiştir.

3. ARAŞTIRMA TASARIMI

Araştırma yöntemi, deneysel verilerin toplandığı ve analiz edildiği bir dizi düzenleyici ilke olarak tanımlanır [7]. Herhangi bir araştırma problemi çeşitli yöntemler ile ele alınabilir. Araştırma tasarımı, belirli bir araştırma problemi için bir yöntem seçme sürecidir. Bu bölümde araştırmanın amacı, araştırma yöntemi, araştırma soruları ve uygulama seçimi anlatılmaktadır.

3.1 Araştırmanın Amacı

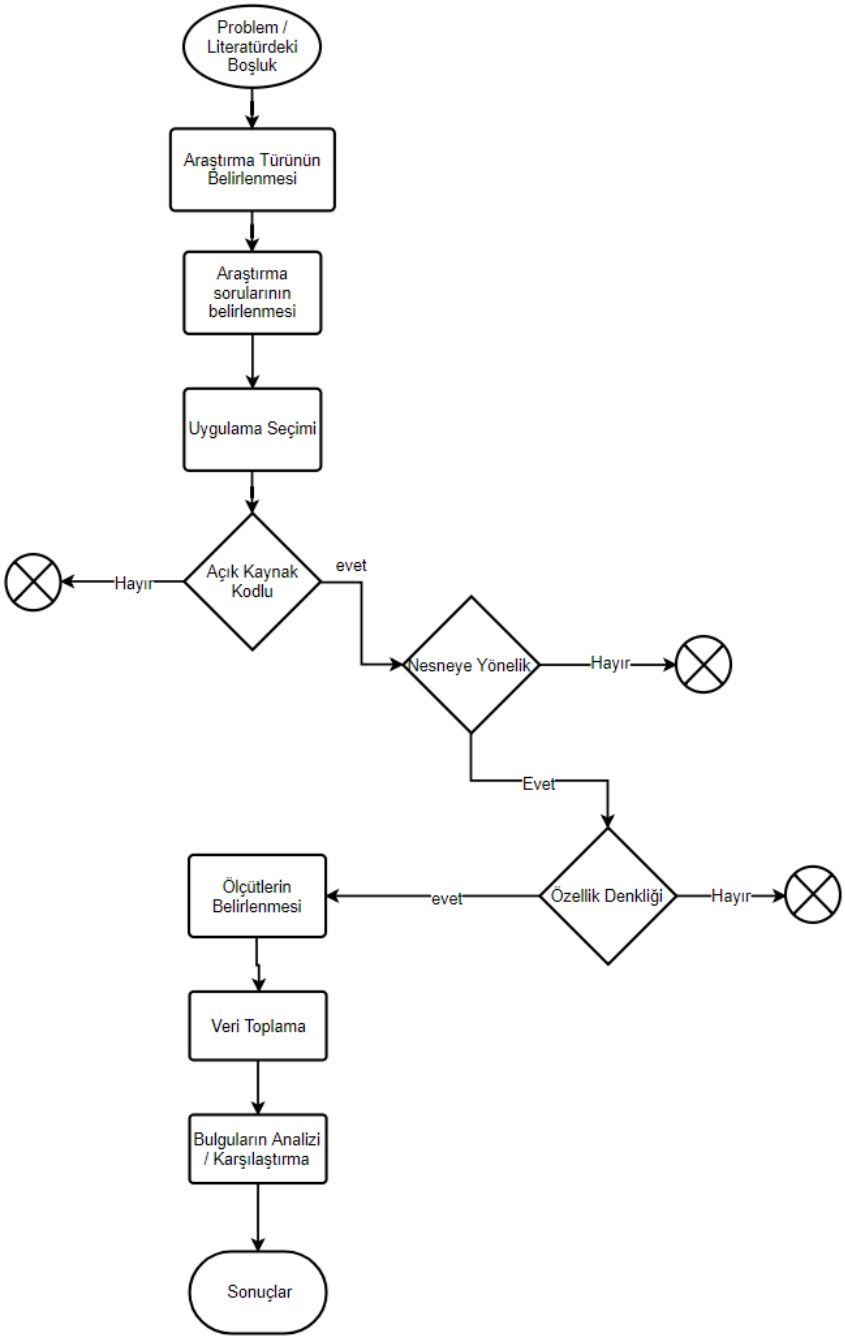
Bu tez çalışmasında, mobil ve masaüstü platformlar için geliştirilen eş özellikler içeren uygulamaların nesneye yönelik tasarımı, kaynak kod büyüklüğü ve platform bağımlılığı incelenmiş, bulgulara ve araştırma sırasında elde edilen tecrübelere dayanarak bu iki platform için geliştirilmiş uygulamalar arasında benzerlik gözetilip gözetilemeyeceğini araştırmak amaçlanmıştır.

3.2 Araştırma Yöntemi

Bu tez çalışmasında araştırma yöntemi olarak keşifsel araştırma yöntemi tercih edilmiştir. Önbilgi bölümünde keşifsel araştırma başlığında (Bölüm 2.1) bahsedildiği üzere, bir konuda belirsiz bir problem olduğunda, o alanda var olan sonuçlar olgunlaşmamış veya kesin değilse ve araştırma süreci tanımlanmamış ya da erken aşamalarda ise keşifsel araştırma yöntemi önerilmektedir [7]. Mobil ve masaüstü uygulamaların geliştirici bakış açısıyla ele alındığı çalışmaların azlığı ve hipotez yetersizliği bu alanda yapılacak bir araştırmanın keşifsel araştırma yöntemi ile yapılmasını elverişli kılmaktadır. Bu nedenle keşifsel araştırma yöntemi tercih edilmiştir.

Şekil 2'de araştırma yönteminin aşamaları verilmektedir. İlk aşamada, araştırmanın amacı ve araştırma soruları belirlenmiştir. Sonrasında, gerçek uygulamaların incelenmesi ile belirlenen araştırma sorularının cevaplanması planlanmıştır. Bu amaçla uygulama seçimi için kriterler belirlenmiştir. Öncelikle açık kaynak kodlu uygulamalar arasında, mobil ve masaüstü platformlar için geliştirilmiş sürümleri olan, denk özelliklere sahip uygulamalar araştırılmıştır. Bulunan uygulamalar arasından ise karşılaştırmanın daha sağlıklı olması açısından, nesneye yönelik dillerle geliştirilmiş olanlar seçilmiştir. İncelenecek uygulamalar seçildikten sonra araştırma sorularını cevaplayacak doğru ölçütler ve

yöntemler belirlenip veri toplama aşamasına geçilmiştir. Son olarak bulguların analizi ve karşılaştırması ile araştırma çalışması tamamlanmıştır.



Şekil 2. Araştırma Yöntemi

3.3 Araştırma Soruları

Bu tez kapsamında belirlenen dört araştırma sorusu (AS) aşağıdadır:

- AS1. Mobil ve masaüstü uygulamaların tasarımı, nesneye yönelik ölçütler bakımından nasıldır?

- AS2. Mobil ve masaüstü uygulamaların büyüklüğü, kaynak kod boyutu bakımından nasıldır?
- AS3. Mobil ve masaüstü platformlar için geliştirilmiş uygulamalar, farklı türden bağımlılıklar açısından nasıldır?
- AS4. Bu iki platform için geliştirilmiş uygulamalar arasında benzerlik gözetilebilir mi?

Bu bölümde her araştırma sorusu için motivasyon ve cevaplamak için belirlenen yöntem açıklanmıştır.

3.3.1 AS1. Mobil ve Masaüstü Uygulamaların Tasarımı, Nesneye Yönelik Ölçütler Bakımından Nasıldır?

Bir yazılımın tasarımı kod seviyesinde ölçülebilmektedir. Bu araştırma sorusu ile uygulamaların masaüstü ve mobil sürümlerinin yazılım tasarım özelliklerinin nasıl olduğu, yazılım tasarım ölçütleri için elde edilen sonuçlar üzerinden incelenmiş ve benzerlik veya farklılıklar ele alınmıştır. Tez çalışmasında incelenen uygulamalar nesneye yönelik dillerle geliştirildiği için, nesneye yönelik ölçüt kümelerinden biri seçilip uygulama çiftleri için ölçütlere ait değerler elde edilmiştir.

Bölüm 2.3'te bahsedildiği üzere, literatürde yazılım tasarımını ölçmek için birçok ölçüt kümesi önerilmiştir. Jabangwe ve arkadaşları yaptıkları bir sistematik literatür taramasında [21] C&K ölçüt kümesinin, nesneye yönelik tasarım ölçütleri arasında en çok tercih edilen (99 çalışmanın %79'u) olduğunu belirtmiştir. Yazarlar aynı zamanda, C&K ölçüt kümesinin, MOOD ve QMOOD ölçüt kümelerinden daha iyi performans sergilediklerini raporlamıştır. Bu nedenle bu tez çalışmasında C&K ölçüt kümesi tercih edilmiştir.

C&K ölçüt kümesinde yer alan 6 ölçüt için ölçümler, iki uygulama çifti için yapıldıktan sonra, her uygulama çifti için sonuçlar karşılaştırmalı olarak analiz edilmiştir.

3.3.2 AS2 Mobil ve Masaüstü Uygulamaların Büyüklüğü, Kaynak Kod Boyutu Bakımından Nasıldır?

Kaynak kod boyutu ölçütlerinin, bir yazılım sisteminin karmaşıklığı ile önemli ölçüde ilişkili olduğu kanıtlanmıştır [22]. Bir sistemin karmaşıklığı ve boyutu, bir yazılımı anlamak, geliştirmek ve bakımını yapmak için gereken çabayı ölçmeye yardımcı olur. Bu araştırma sorusu ile geliştiriciler benzer işlevlere sahip bir uygulamayı, mobil ve masaüstü platformlar için geliştirirken kod boyutunun nasıl

değiştiğini ortaya koymak amaçlanmıştır. Buna ek olarak, bu iki platform için geliştirilmiş uygulamaların, üçüncü parti kaynak koda ne kadar bağımlı olduğunu araştırmak da hedeflenmiştir. Üçüncü parti kütüphaneler; geliştirilen uygulamadan ayrı olarak, farklı geliştiriciler tarafından ortaya konmuş, yeniden kullanılabilir yazılım bileşenleridir. Uygulama geliştirilirken, sıklıkla genelde üçüncü parti kütüphanelerden yararlanılır. Bu kütüphaneler, gerektiğinde uygulamaya uygun olacak şekilde özelleştirilir ve bakımları yapılır. Bu nedenle üçüncü parti kütüphanelere ait kaynak kodunun ve projeye özgü kaynak kodunun birbirinden ayrı olarak analiz edilmesi önemlidir [4].

Bu soruyu cevaplarken kod satır sayısı, dosya sayısı ve sınıf sayısı kod boyutu ölçütleri kullanılmıştır. Bu ölçütler için elde edilen değerler ile her bir mobil ve masaüstü uygulama çiftinin kaynak kodu ölçütleri, üçüncü parti kaynak kod ve projeye özel kaynak kod olmak üzere iki alt küme halinde karşılaştırılmıştır.

3.3.3 AS3 Mobil ve Masaüstü Platformlar İçin Geliştirilmiş Uygulamalar, Farklı Türden Bağımlılıklar Açısından Nasıldır?

Bu soru cevaplanırken uygulamaların sınıflarını geliştirme dili, üçüncü parti kaynak kodu, projeye özel kaynak kodu ve platforma özel kaynak koduna olan bağımlılıkları hesaplanıp karşılaştırılmaktadır. Farklı platformlar için benzer uygulamalar geliştirilirken, geliştiricilerin uygulamalarını birden fazla platforma taşıması gerekir. Platform bağımlılığı, bu süreçte yapılması gereken iş miktarını olumsuz yönde etkiler. Platform bağımlılığının düşük olması, geliştiricilerin platform API'larına (İng. Application Programming Interface – Uygulama Programı Arabirimi) önemli ölçüde dayanmadıklarını gösterebilir. Bu tür uygulamaların farklı platformlara taşınması daha kolay olabilir. Öte yandan, platform bağımlılığı yüksek ise platform API'sinin yoğun kullanımının bir platform kilitlenmesine, diğer bir deyişle diğer platformlara geçişin zor ve karmaşık olmasına neden olabilir.

Bu tez çalışmasında, Bölüm 2.4'de anlatıldığı üzere, Syer ve arkadaşlarının çalışmasında [4] önerdiği Platform Bağımlılık Oranı (İng. Platform Dependency Ratio - PDR) ölçütü çalışmaya uygun olarak adapte edilmiş ve kullanılmıştır. Syer ve arkadaşları platform bağımlılığını, platform bağımlılığı ve platforma dayalı şeklinde iki kategoride ele almışken bu çalışmada, bu bağımlılıklar bir arada değerlendirmiş ve platform bağımlılığı olarak ele alınmıştır. Elde edilen sonuçlar, incelenen mobil ve masaüstü uygulama çiftleri için değerlendirilmiştir.

3.3.4 AS4. Bu İki Platform İçin Geliştirilen Uygulamalar Arasında Benzerlik Gözetilebilir mi?

Farklı platformlar için uygulama geliştirirken bir platformun, uygulamanın kaynak kodunu etkileyip etkilemediğini veya etkilediyse nasıl etkilediğini bilmek ve benzerlik ve farklılıkları ortaya koymak, geliştiricilere yardımcı olabilir. Bu noktada, bir uygulamanın kaynak kodunu etkileyen etmenlerin çeşitliliği ve bu etmenlerin tamamı hakkında bilgi sahibi olamamak zorlayıcıdır.

Bu tez çalışmasında, mobil ve masaüstü platformlar için farklı dillerle geliştirilmiş, denk özelliklerde iki uygulama çifti incelenmiştir. İlk üç araştırma sorusu ile bu uygulamaların tasarım ölçütleri, kaynak kod boyutu ve platform bağımlılıkları açısından nasıl olduğu incelenmiştir. Bu araştırma sorusu ile diğer araştırma sorularının cevapları aranırken elde edilen veriler ve araştırmanın yapıldığı koşullar göz önünde bulundurularak; farklı dillerle geliştirilmiş, denk özelliklere sahip mobil ve masaüstü uygulamaların, platform karşılaştırması yapmak için elverişli olup olmadığının değerlendirilmesi amaçlanmıştır.

3.4 Uygulama Seçimi

Araştırma sorularının cevaplarını bulmak amacıyla çalışmada, farklı platformlar için denk özelliklere sahip mobil uygulama ve masaüstü uygulama çiftleri incelenmek üzere seçilmiştir. Seçilen uygulamalar için belirlenen kriterler şöyledir:

- *Açık Kaynak Kodlu Uygulamalar.* Kod tabanlı bir araştırma yapıldığı için uygulamaların kod havuzlarına erişilmesi gerekmektedir. Açık kaynak kodlu uygulamaların GitHub gibi kod depolarından kaynak kodlarına erişilebilmektedir.
- *Nesneye yönelik programlama dilleriyle geliştirilmiş uygulamalar (Java, C++).* Geleneksel nesne odaklı ölçüt kümelerinin kullanılabilmesi için nesneye yönelik bir dil ile geliştirilmiş uygulamalar tercih edilmiştir. Bu nedenle, masaüstü sürümleri C dilinde geliştirilmiş uygulamalar hariç tutulmuştur.
- *Uygulamaların Özellik Denkliği.* Mobil ve masaüstü uygulamalar arasında daha nesnel bir karşılaştırma yapabilmek için, aynı uygulamanın birbirine denk özellikler içeren mobil ve masaüstü sürümlerinin olması gerekmektedir.

Yukarıdaki kriterlere ek olarak; küçük kod hacmine sahip, az katkıda bulunulan ve az sayıda kod işlemi içeren uygulamalar değerlendirmeye dâhil edilmemiştir.

Uygulamalarda özellik denkliği ve geliştirme dilinin nesneye yönelik olması kriterlerinin aranması, uygulama havuzunu daraltan baskın ölçütlerdir. Bu kriterler uygulandıktan sonra iki uygulama çifti seçilmiştir: Telegram ve KeePass. Uygulamaların mobil sürümlerinin tümü, Android platform için geliştirilmiştir. Uygulamaların kaynak kodu depolarına Nisan 2017'de ulaşılmıştır. Seçilen uygulamaların ayrıntıları Tablo 1'de verilmiş ve takip eden alt başlıkta bu uygulamalar kısaca tanıtılmıştır.

Tablo 1. Platform Karşılaştırması için Seçilen Uygulamalar

Uygulama	Gel. Dili	Kaynak kod web adresi	İncelenen Sürüm Bilgisi
KeePass	C++	https://github.com/keepassx/keepassx	v 2.0.3
KeePassDroid	Java	https://github.com/bpellin/keepassdroid	v 2.0.6.4
Telegram Masaüstü	C++	https://github.com/telegramdesktop/tdesktop	v 1.1.0
Telegram Mobil	Java	https://github.com/DrKLO/Telegram	v.0

3.4.1 İncelenen Uygulamalar

***KeePass*:** KeePass şifrelerin güvenli bir şekilde yönetilmesine yardımcı olan ücretsiz bir açık kaynak uygulamadır [22]. Java ile geliştirilmiş Android sürümü ve C++ ile geliştirilmiş masaüstü sürümü bulunmaktadır. KeePass'ın temel özellikleri aşağıda verilmektedir:

- Güçlü güvenlik
- Çoklu kullanıcı şifresi depolama
- Taşınabilirlik, kurulum gerektirmeme
- Kolay veri tabanı transferi
- Şifre gruplama desteği
- Otomatik tamamlama ve sürükle-bırak özelliği
- Arama ve sıralama
- Eklenti mimarisi
- Çoklu dil desteği
- Güçlü rastgele şifre üretici
- TXT, HTML, XML ve CSV dosya türlerinde dışarı veri aktarımı
- Birçok farklı dosya türünden veri alma

Tez çalışması ile incelenen KeePass uygulamasının masaüstü sürümü C++ ile geliştirilmiştir ve 109 sınıf, 194 dosya ve 18.438 Kod Satır Sayısı (KSS – İng. LOC)'ndan oluşmaktadır. KeePass uygulamasının mobil sürümü ise Java programlama dili ile geliştirilmiştir ve 291 sınıf, 204 dosya ve 13.953 KSS'den oluşur.

***Telegram*:** Telegram, hız ve güvenliğe odaklanan açık kaynaklı bir mesajlaşma uygulamasıdır [23]. Telegram'ın Android, iOS ve WP için mobil sürümleri; PC, Mac ve Linux için masaüstü sürümleri; ayrıca çapraz platform uygulaması olarak bir web sürümü bulunmaktadır [23]. Öne çıkan özellikleri şöyledir:

- Mesajların uçtan uca şifrelenmesi,
- Bir zamanlayıcı ile mesajları otomatik imha edilebilmesi,
- Medya ve sohbetlerin boyutlarında kısıtlama olmaması,
- Bulut ortamında medya içeriklerinin depolanması,
- Farklı cihazlardan sohbetlerin senkronize edilebilmesi.

Tez çalışmasında Telegram'ın C++ ile geliştirilen masaüstü sürümü ve Java ile geliştirilen Android sürümü incelenmiştir. Masaüstü sürüm 1435 sınıf, 515 dosya ve 220.986 KSS'den oluşmaktadır. Android sürüm ise 4472 sınıf, 839 dosya ve 450.952 KSS'den oluşur.

4. SONUÇLAR

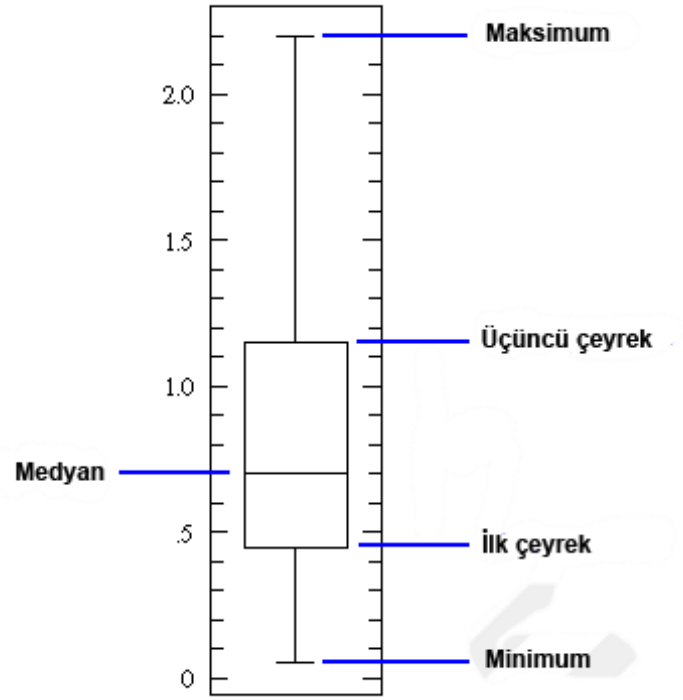
4.1 AS1 Mobil ve Masaüstü Uygulamaların Tasarımı, Nesneye Yönelik Ölçütler Bakımından Nasıldır?

Bu araştırma sorusu ile uygulamaların masaüstü ve mobil sürümlerinin yazılım tasarım ölçütleri bakımından nasıl olduğunu ortaya koymak hedeflenmiştir. Bölüm 3.3.1'de belirtildiği üzere C&K ölçüt kümesi seçilmiştir. İki uygulama çiftine ait sınıflar için, C&K ölçüt kümesinde bulunan altı ölçütün değerleri hesaplanmıştır. Bu hesaplamalar yapılırken Understand aracından yararlanılmıştır. Understand aracının C&K ölçütlerini nasıl tanımladığı Tablo 2'de verilmektedir.

Tablo 2. Understand Aracına göre C&K Ölçütleri

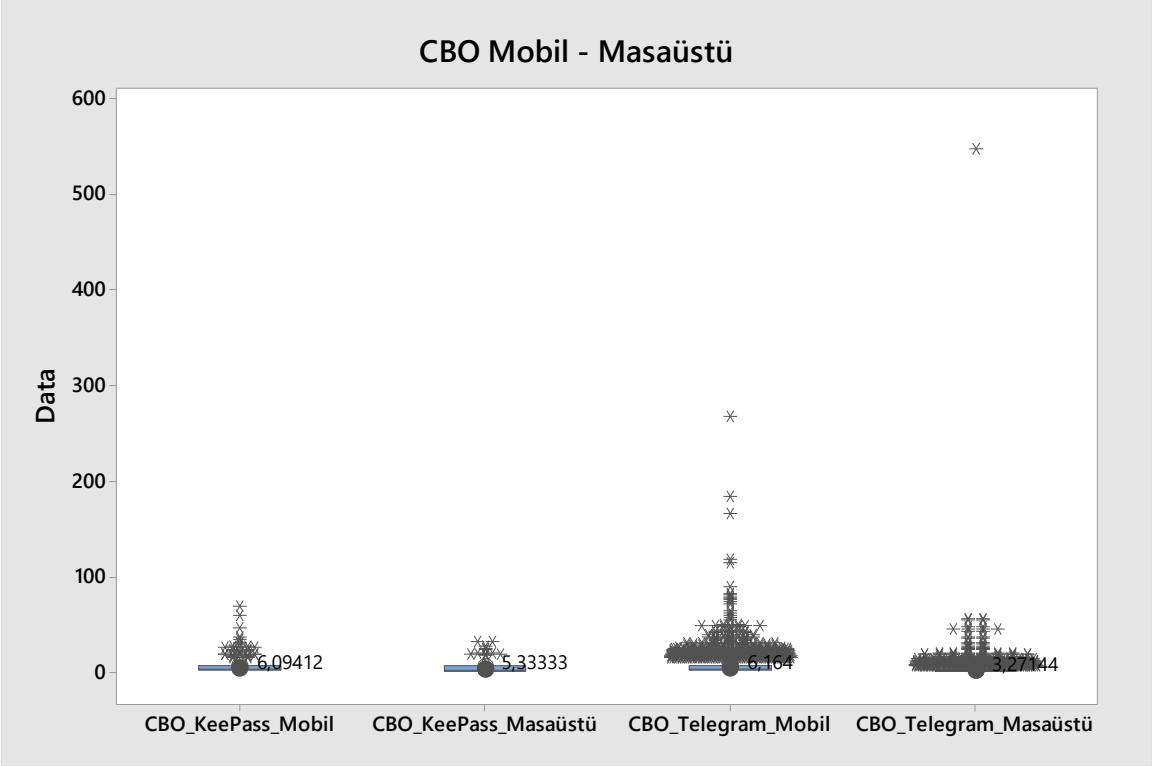
C&K Ölçütü	**Understand Aracında Tanımlı İsim**	**Tanımlama**
CBO	CountClassCoupled	Bağlı olunan sınıfların sayısı (Bir A sınıfı B sınıfından bir tür, veri veya üye eleman kullanıyorsa, A sınıfı B sınıfına bağlıdır.)
NOC	CountClassDerived	Anlık alt sınıf sayısı (Kalıtım ağacında, bu sınıftan bir alt seviyede olan sınıfların sayısı)
DIT	MaxInheritanceTree	Sınıfın kalıtım ağacındaki maksimum derinliği (Kök düğümü 0 (sıfır) kabul edilir.)
LCOM	PercentLackOfCohesion	100% - Sınıf değişkenlerinin ortalama uyum değeri. (Sınıf yöntemlerinin yüzde kaçının belirli bir sınıf değişkeni kullandığını hesaplar. Tüm metotlar için elde edilen değerlerin ortalaması değer 100%'den çıkarılır.)
RFC	CountDeclMethodAll	Kalıtımlarla gelen metotlar dâhil toplam metot sayısı
WMC	CountDeclMethod	Yerel metotların sayısı

Ölçütler için elde edilen sonuçlar kutu grafiği biçiminde verilmiş ve yorumlanmıştır. Kutu grafiği bir veri setine ait beş sayının özetine dayanan bir veri gösterim biçimidir [24]. Bu beş değer şöyledir: minimum, ilk çeyrek, medyan, üçüncü çeyrek ve maksimum (Bkz. Şekil 3). Verilerin dağılımını göstermek için kullanılır.

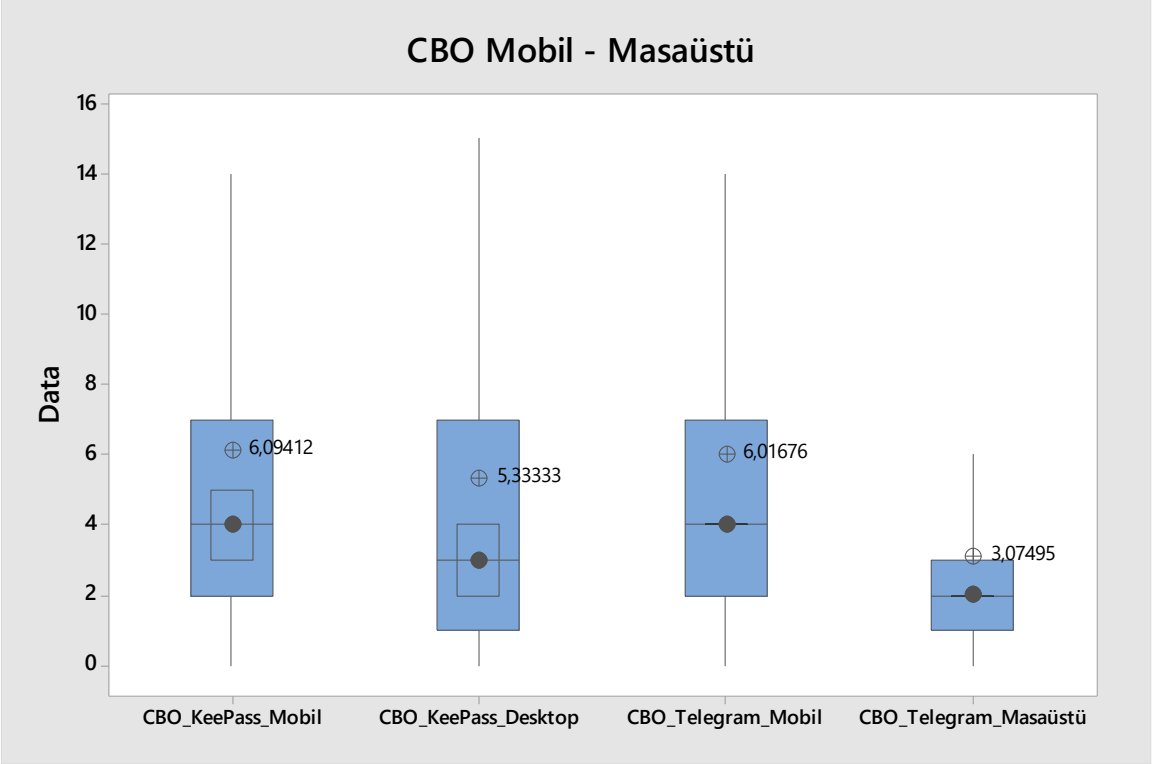


Şekil 3. Kutu Grafiği Gösterimi

CBO, sınıfın bağımlı olduğu sınıfların sayısı olarak tanımlanır. Nesne sınıfları arasındaki aşırı bağlanma, modüler tasarıma zarar verir ve yeniden kullanılabilirliği olumsuz etkiler [13]. Bir sınıf, bağımsızlığı arttıkça, başka bir uygulamada tekrar kullanılmak için daha elverişli hale gelir. Bu nedenle, nesne sınıfları arası bağımlılığın düşük olması tercih edilir. Şekil 4 ve Şekil 5'te KeePass ve Telegram'ın mobil ve masaüstü sürümleri için elde edilmiş nesne sınıfları arasındaki bağımlılık (CBO) değerleri, kutu grafiği şeklinde görülmektedir. Grafiklerden ilki tüm verilerin kullanılmasıyla, ikincisi ise dağılımda tespit edilen aykırı değerlerin (İng. outlier) veri kümesi dışına alınması ile elde edilmiştir. Ek olarak, ikinci grafikte verilerin %95'inin yer aldığı aralık gösterilmektedir. Uygulama çiftlerinde ortalama ve medyan değerlere bakıldığında, her iki uygulama çifti için de mobil sürümlerde sınıflar arası bağımlılığın daha yüksek olduğu görülmektedir.



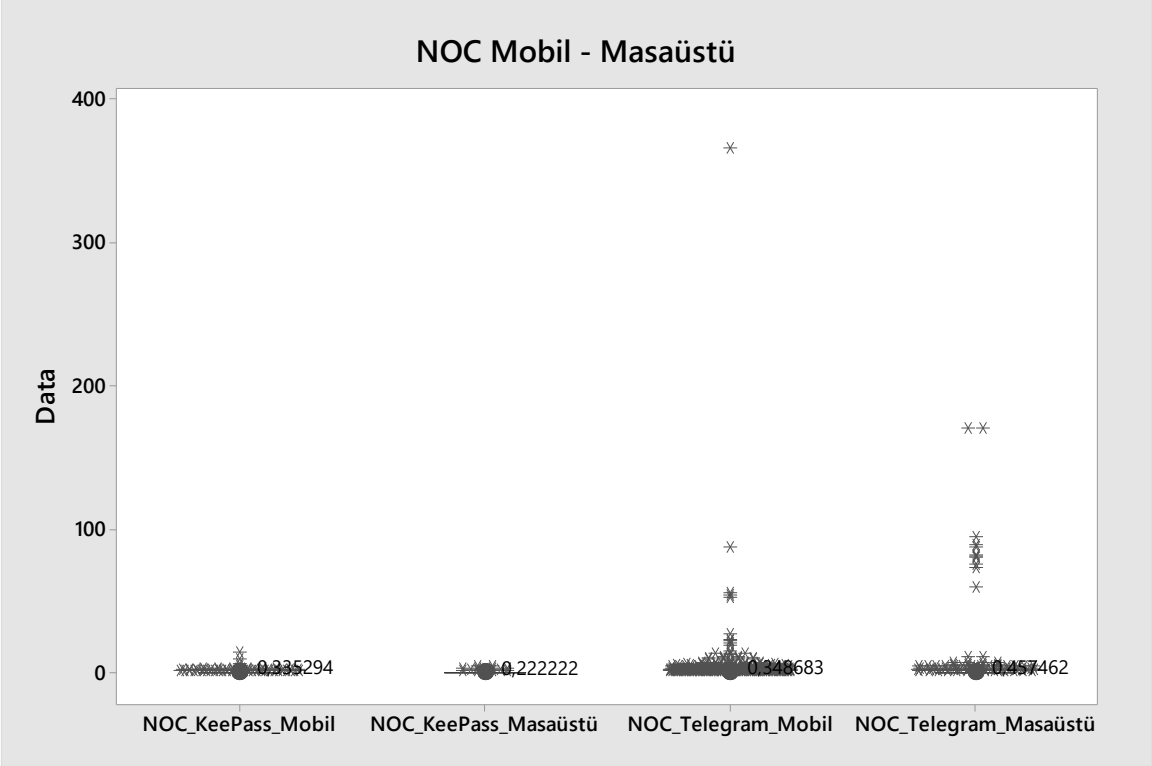
Şekil 4. Nesne sınıfları arasındaki bağımlılık kutu grafiği - mobil ve masaüstü



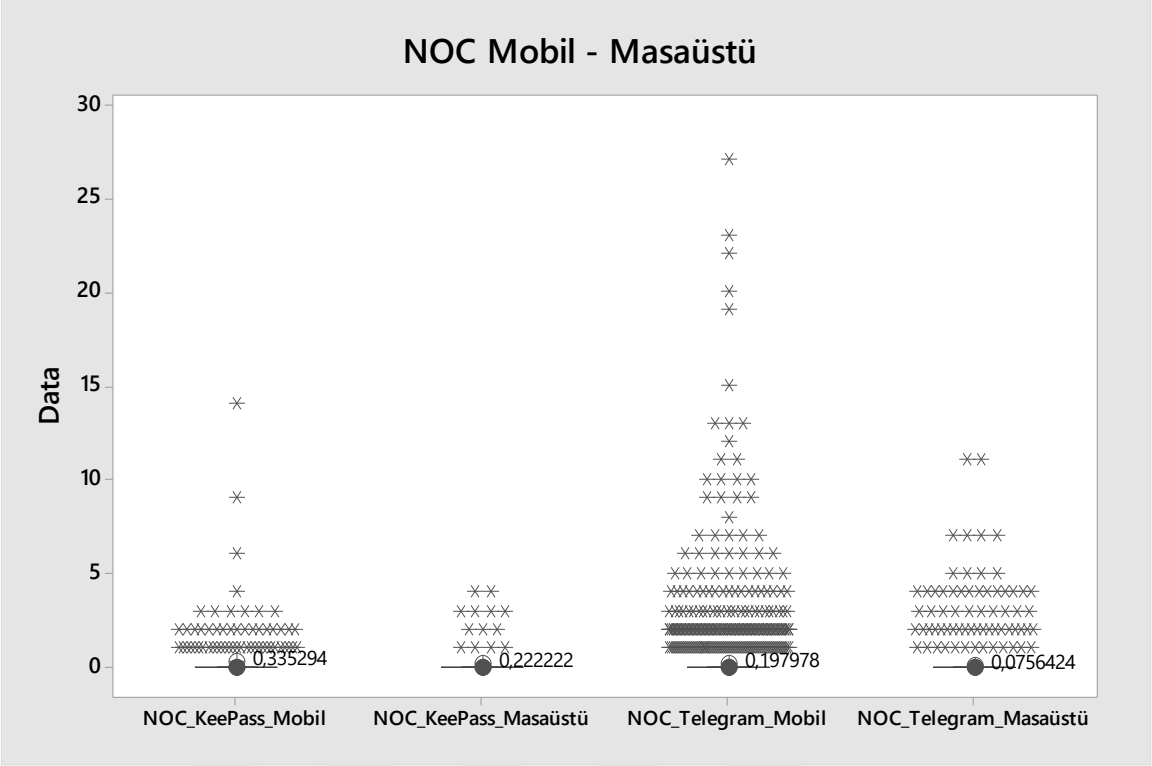
Şekil 5. Nesne sınıfları arasındaki bağımlılık kutu grafiği (aykırı değerler hariç) - mobil ve masaüstü

NOC, bir temel sınıftan türetilen anlık çocuk sınıflarının sayısına eşittir. Çok sayıda çocuğun bulunduğu sınıfların değiştirilmesinin zor olduğu ve genellikle tüm

çocuklarda meydana gelen değişikliklere bağlı olarak daha fazla test yapılması gerektiği düşünülmektedir [25]. Şekil 6 ve Şekil 7'da, her iki uygulama çifti için elde edilen alt sınıf sayısı (NOC) ölçütü değerleri, kutu grafiği biçiminde görülmektedir. Grafiklerden üstteki tüm verilerin kullanılması ve aykırı değerlerin gösterilmesi ile, alttaki ise aykırı değerlerden bazılarının veri kümesi dışına alınması ile elde edilmiştir. Her uygulamada, sıfır değerine sahip çok sayıda sınıf bulunduğu için güven aralığı grafikte görülememektedir. Dağılımların net bir biçimi olmamakla birlikte ortalama değerlere bakıldığında, KeePass uygulama çifti için mobil sürümün ve Telegram uygulama çifti için ise masaüstü sürümün daha büyük değere sahip olduğu görülmektedir.



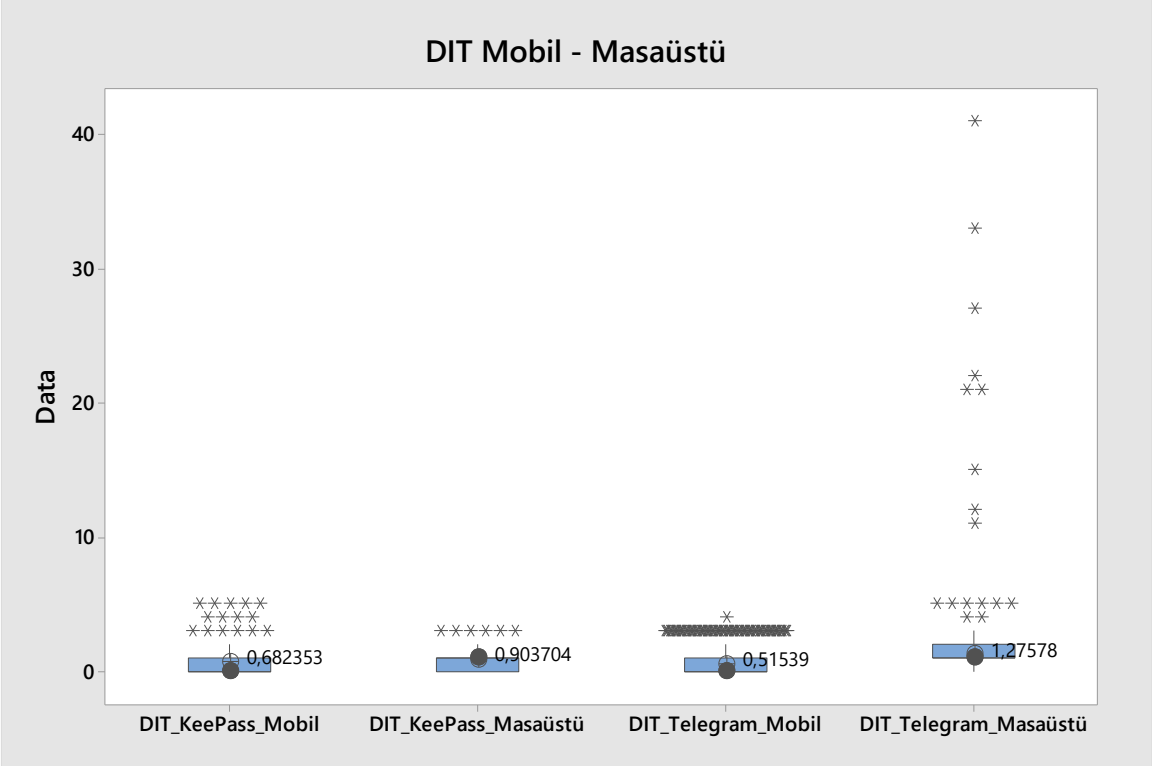
Şekil 6. Alt sınıf sayısı kutu grafiği - mobil ve masaüstü



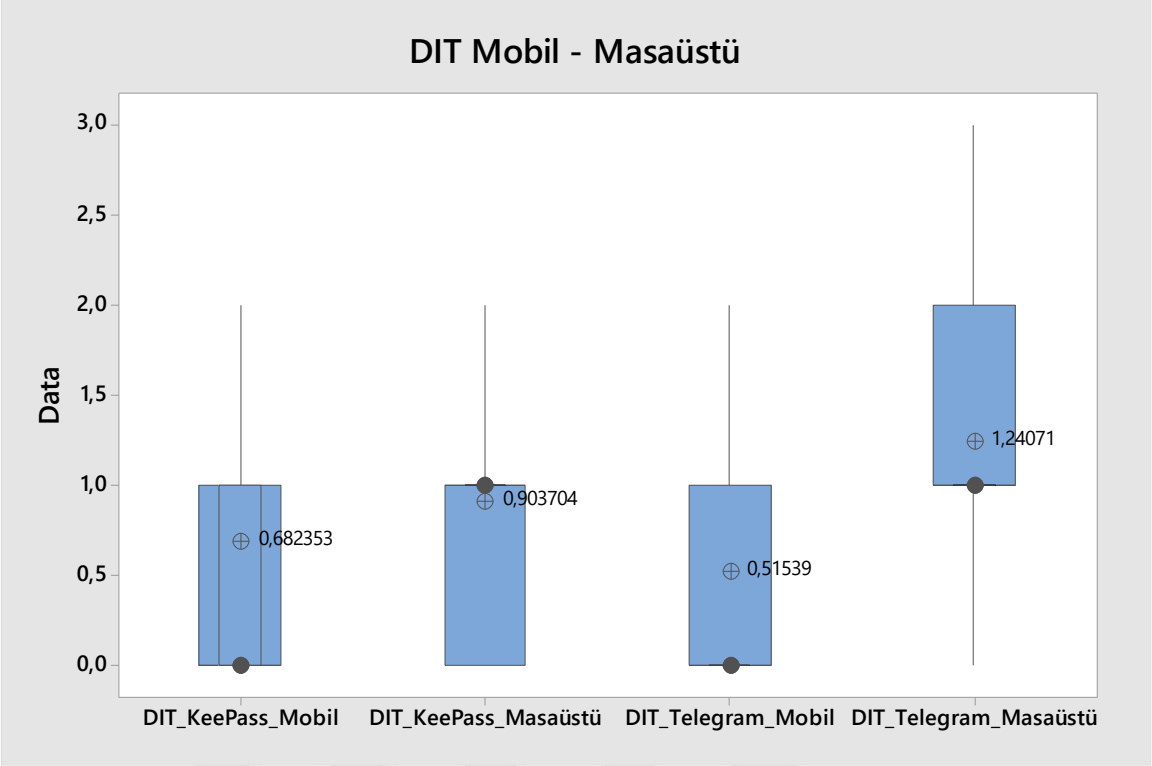
Şekil 7. Alt sınıf sayısı kutu grafiği (aykırı değerler hariç) - mobil ve masaüstü

Bir sınıfın kalıtım hiyerarşisindeki derinliği, sınıf düğümünden sınıf hiyerarşisi ağacının köküne kadar olan maksimum uzunluk olarak tanımlanır ve ata sınıflarının sayısıyla ölçülür. Bir sınıf, hiyerarşide ne kadar derin olursa, kalıtımdan gelen yöntemlerin sayısının o kadar büyük olması beklenir ve bu, sınıfın karmaşıklığını artırır [9]. Düşük kalıtım derinliğine sahip sınıfların, daha yüksek kalıtım derinliğine sahip sınıflara kıyasla bakım ve değişiklik yapma bakımından daha fazla kolaylık sağladığı ve aynı zamanda bir kalite unsuru olan anlaşılırlığının daha yüksek olduğu raporlanmıştır [26]. Şekil 8 ve Şekil 9'de KeePass ve Telegram uygulama çiftindeki sınıflar için hesaplanan kalıtım hiyerarşisindeki derinlik (DIT) ölçütü için değerler, kutu grafiği ile gösterilmiştir. Grafiklerden ilki tüm verilerin kullanılmasıyla, ikincisi ise dağılımda aykırı verilerin veri kümesi dışına alınması ile elde edilmiş ve %95 güven aralığı çizdirilmiştir. KeePass uygulama çifti için ilk grafiğe bakıldığında, mobil sürüm tek başına daha yüksek değerlere sahip olsa da ortalama olarak masaüstü sürümün daha yüksek bir kalıtım ağacı derinliğine sahip olduğu görülmektedir. İkinci grafiğe bakıldığında, medyan mobil sürümde sıfır iken masaüstü sürümde bir (1) olduğu, ayrıca her iki uygulamanın sınıflarına ait DIT ölçütü değerlerinin %95'inin 0-1 aralığına düştüğü görülmektedir. Telegram uygulama çifti için ilk grafiğe bakıldığında, mobil sürümün çoğu sınıfının

kalıtım ağacı derinliğinin bir (1) olduğu, masaüstü sürümün ise daha yüksek değerlere sahip aykırı sınıfları olduğu görülmektedir. Bu aykırı değerlerin çıkartıldığı ikinci grafiğe bakıldığında, masaüstü sürümün sınıflarının %95'inin DIT ölçütü değerinin 1 ve 2 olduğu, mobil sürümün ise 0 ve 1 olduğu görülmektedir. Ortalama değere bakıldığında, masaüstü sürümün kalıtım ağacı derinliği mobil sürümden daha fazladır. Her iki uygulama çifti için de masaüstü sürümlerin DIT değerinin daha büyük olduğu görülmektedir.

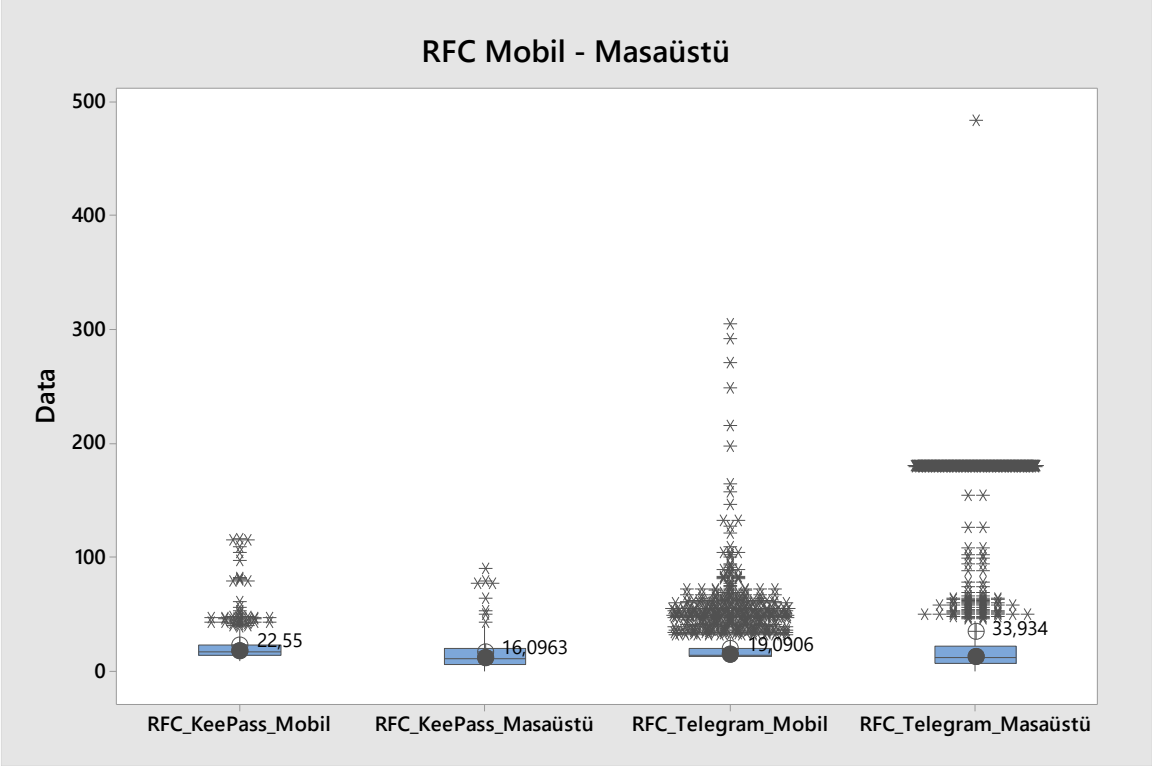


Şekil 8. Kalıtım ağacının derinliği kutu grafiği - Mobil ve Masaüstü

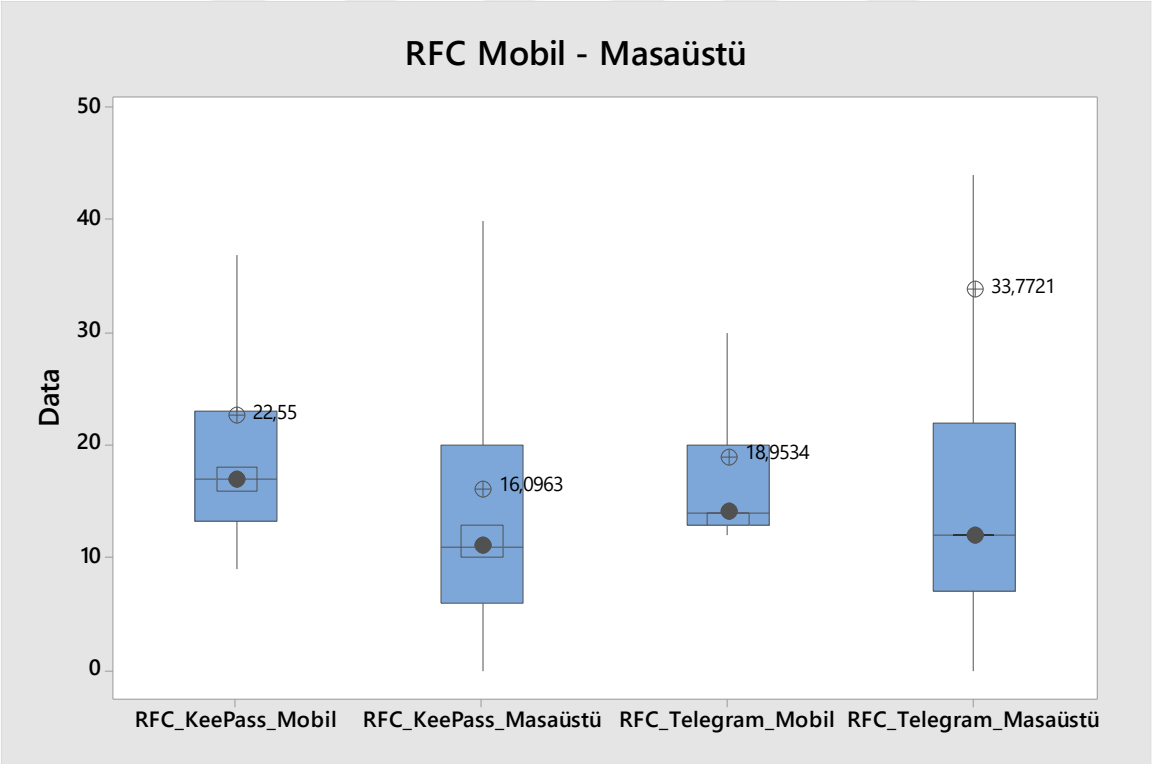


Şekil 9. Kalıtım ağacının derinliği kutu grafiği (aykırı değerler hariç) - mobil ve masaüstü

Bir sınıfın tetiklediği metot sayısı, o sınıfın bir nesnesi tarafından alınan bir mesaja yanıt olarak tetiklenebilen toplam metot sayısıdır [9]. RFC, sınıfın davranışıyla ilgili bilgi verir. Yüksek bir RFC değerine sahip sınıflar daha karmaşıktır ve bu sınıfların anlaşılması daha zordur. Test etme ve hata ayıklama karmaşıktır. Aynı zamanda hataya yatkınlık daha fazladır. Şekil 10 ve Şekil 11'da KeePass ve Telegram uygulama çiftleri için elde edilen, sınıfların tetiklediği metot sayıları (RFC), kutu grafiği ile verilmektedir. İlk grafikte görülen renkli çerçeve, istatistiksel olarak verilerin %25 ile %75'inin yer aldığı alanı ifade etmektedir. İkinci grafikte ise, verilerin %95'inin yer aldığı aralığı gösteren güven aralığı görülmektedir. KeePass uygulama çifti için bu aralıkların her iki grafikte de mobil sürümde az bir farkla daha dar ve yukarıda olduğu, ortalama değerin ise mobil sürüm için 22,55 ve masaüstü sürüm için 16,09 olduğu görülmektedir. Telegram uygulama çifti için grafiklere bakıldığında, mobil sürümün tek başına RFC değeri daha yüksek sınıflara sahip olmasına rağmen, masaüstü sürümün ortalama değerinin mobil sürümden oldukça büyük olduğu görülmektedir. Buna sebep olan, RFC değeri yüksek olan bir grup sınıfın bulunmasıdır. Ayrıca %95 aralığına bakıldığında ve her iki uygulama çifti bir arada değerlendirildiğinde mobil sürümlerin, masaüstü sürümlere göre daha büyük RFC değerlerine sahip olduğu söylenebilir.



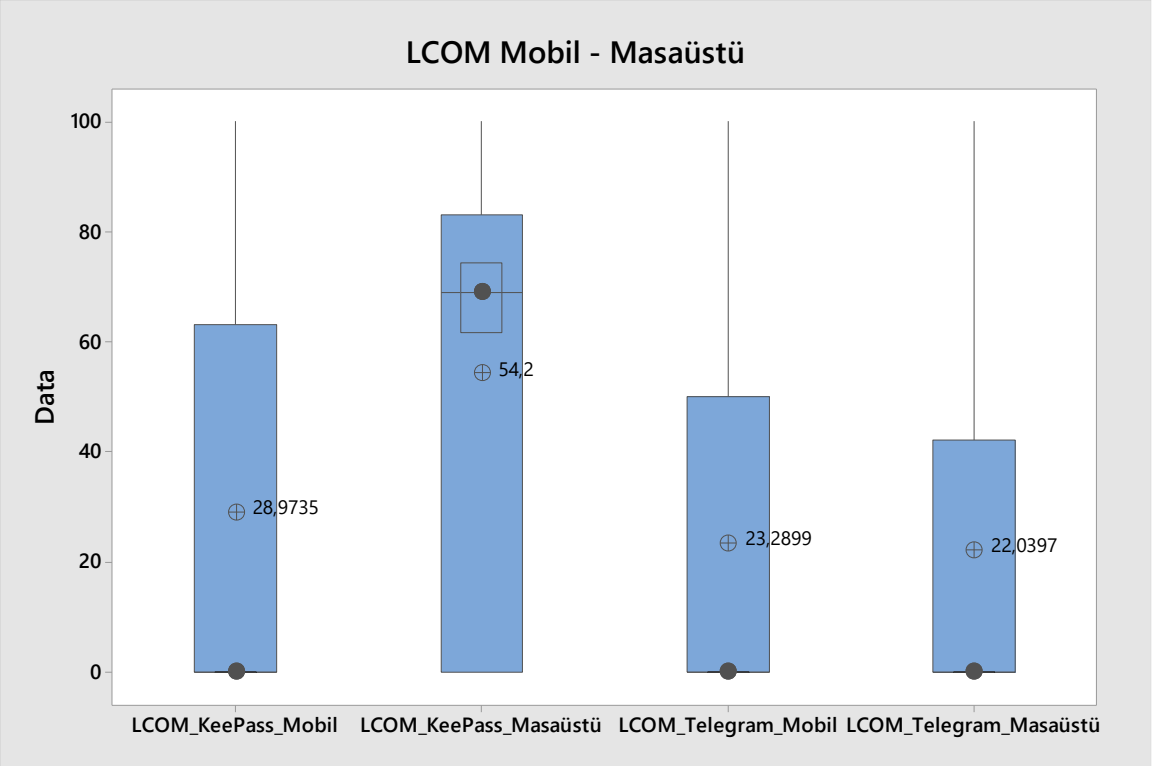
Şekil 10. Sınıfın tetiklediği metot sayısı kutu grafiği - mobil ve masaüstü



Şekil 11. Sınıfın tetiklediği metot sayısı kutu grafiği (aykırı değerler hariç) - Mobil ve Masaüstü

Metotlardaki uyum eksikliği (LCOM), metotlardaki benzerlik derecesini ölçer [9]. Bu ölçüt değeri kullanılarak yazılımın verimlilik ve yeniden kullanılabilirlik gibi

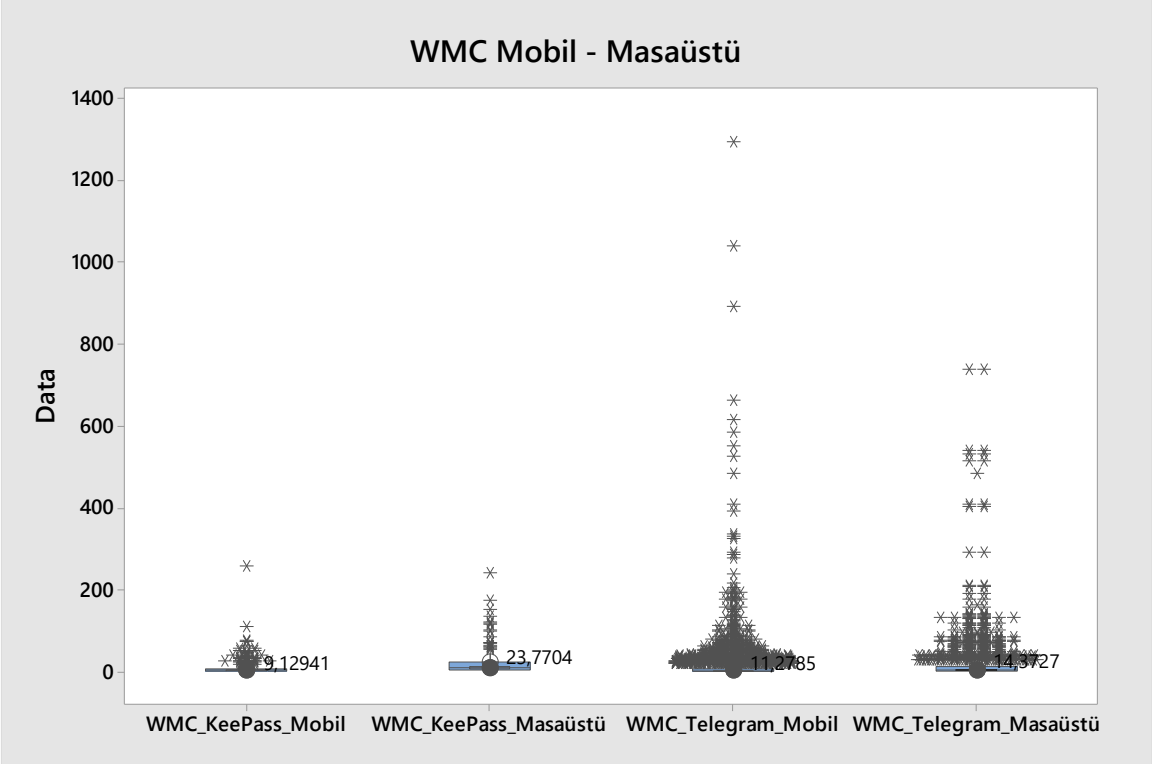
değerlerinin ölçülmesi mümkündür. Uyum eksikliği veya düşük bağlılık karmaşıklığı artırır, böylece geliştirme süreci sırasında hataların olasılığı artar. Şekil 12'de uygulamalara ait sınıfların metotlardaki uyum eksikliği ölçütü için değerleri, kutu grafiği biçiminde görülmektedir. Telegram'ın masaüstü ve mobil sürümünün ortalama LCOM değerleri ve değerlerin dağılımı, birbirinden önemli ölçüde farklılık göstermemektedir. Az farkla mobil sürümün ortalaması daha yüksek ve dağılım aralığı daha geniştir. KeePass uygulama çiftine bakıldığında ise masaüstü sürümün LCOM değerlerinin hem dağılım hem de ortalama değer bakımından daha yüksek olduğu görülmektedir.



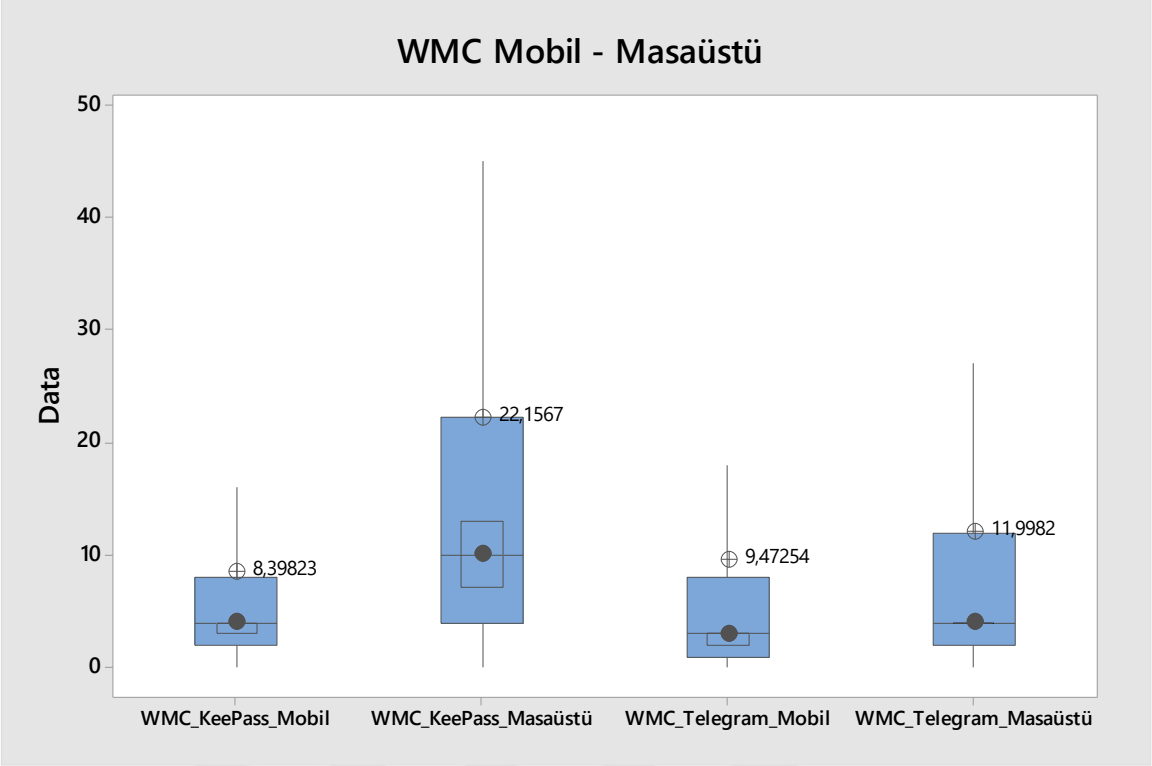
Şekil 12. Metotlardaki uyum eksikliği kutu grafiği - mobil ve masaüstü

WMC, bireysel bir sınıfın karmaşıklığını ölçer. Çok sayıda yöntem içeren sınıfların, daha fazla uygulamaya özgü olmasının, yeniden kullanma olasılığını sınırladığı söylenir . WMC, sınıfı geliştirmek ve bakımını yapmak için ne kadar zaman ve çaba gerektiğini öngörmeyi sağlar. Yeniden kullanılabilirlik ve bakım maliyetleri göz önünde bulundurulduğunda, WMC'nin genellikle düşük olması tercih edilir. Şekil 13 ve Şekil 14'de uygulamalara ait sınıfların ağırlıklı metot sayısı, kutu grafiği biçiminde görülmektedir. İlk grafikte, verinin %25 ve %75'inin yer aldığı aralığı ifade eden alt çeyrek ve üst çeyrek sınırlar görülmektedir. İkinci grafik, gözle tespit edilen aykırı değerlerin çıkartılmasıyla elde edilmiştir ve verinin %95'inin yer aldığı

güven aralığı görülmektedir. Grafiklere birlikte bakıldığında her iki uygulama çifti için, masaüstü sürümün daha geniş bir aralığa ve değere sahip olduğu söylenebilir. KeePass'ın mobil uygulamasına ait ortalama 9,13 iken masaüstü uygulamaya ait ortalama 23,77'dir. Telegram'ın mobil uygulaması için ortalama 11,28 ve masaüstü uygulama için ortalama 14,37 olarak hesaplanmıştır. KeePass uygulaması için iki sürümün ortalama WMC değerleri arasındaki fark daha yüksek olsa da her iki uygulama çifti için, masaüstü sürümlerin daha büyük WMC değerlerine sahip olduğu gözlemlenmiştir.



Şekil 13. Sınıfın ağırlıklı metot sayısı kutu grafiği - mobil ve masaüstü



Şekil 14. Sınıfın ağırlıklı metot sayısı kutu grafiği (aykırı değerler hariç) - mobil ve masaüstü

4.2 AS2 Mobil ve Masaüstü Uygulamaların Büyüklüğü, Kaynak Kod Boyutu Bakımından Nasıldır?

Bu araştırma sorusu ile geliştiricilerin benzer işlevlere sahip bir uygulamaları, Android platformu için Java ve masaüstü platform için C++ ile geliştirirken kod boyutunun nasıl değiştiğini ortaya koymak amaçlanmıştır. Bu amaçla üç kaynak kod boyutu ölçütü seçilmiştir; kod satır sayısı, dosya sayısı, sınıf sayısı. Bu ölçütlerle ilgili verileri çıkarmak için SciTools'un Understand aracından yararlanılmıştır. Kod satır sayısı çıkartılırken Understand aracında tanımlı CountLineCode ölçütü kullanılmıştır. Kaynak kod içeren satırların sayılmasıyla elde edilen bu ölçümde, yorum satırları hariç tutulmuştur.

Hesaplanan kod satır sayısı, dosya sayısı ve sınıf sayısı ölçütleri üç şekilde incelenmiştir;

- Toplam kaynak kodu boyutu,
- Projeye özel kaynak kodu boyutu,
- Üçüncü parti kütüphanelerin kaynak kodu boyutu.

Üçüncü parti kütüphanelerin kullanımı, geliştirme yükünü ve dolayısıyla geliştirme süresini azaltmaktadır; buna karşın, bakım gereksinimlerine sebep olabilir ve daha

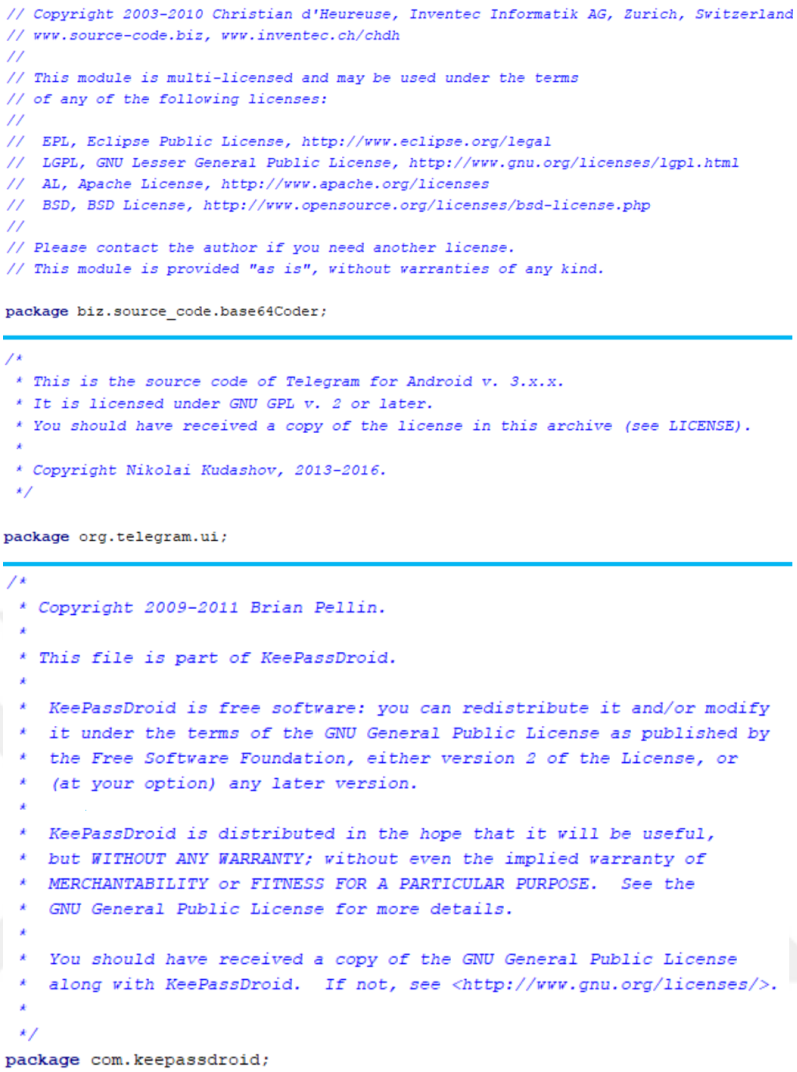
fazla bağımlılık yaratabilir. Proje için uygulamaya özel olarak geliştirilmiş kodun miktarı ise bir uygulama için gereken geliştirici çabasının bir göstergesi olduğu için oldukça önemlidir.

Her uygulama için, hangi kaynak kod dosyalarının üçüncü tarafa ait olduğu ve hangi dosyaların projeye özel kod olduğu tespit edilmiştir. Uygulamaların dosya ağaçları taranmış ve projeye özel ve üçüncü parti kütüphanelere ait olabilecek alt bölümler tespit edilmiştir. Genellikle kod dosyalarının en üst kısmında yer alan bölümde, o kaynak kodun kim tarafında geliştirildiği bilgisi yer almaktadır (Bkz. Şekil 15). Bu bilgiden de yararlanarak kaynak kodlar projeye özel ve üçüncü parti olarak ayırt edilmiştir.

KeePass mobil ve masaüstü sürümlerinin kod tabanları, test kodu ve test koduyla ilgili üçüncü parti kütüphaneleri içermektedir. Bu dosyalar, uygulamanın çalışması için gerekli olmadığından kaynak kodu boyutu incelemesinde hariç tutulmuştur.

Tablo 3. Toplam Kaynak Kodu Boyut Ölçütleri ve Platforma Göre Farklılıkları

Telegram			
Ölçüt	Masaüstü sürümü	Mobil sürümü	Fark oranı (%)
# Dosya	515	839	+%63
# Sınıf	1435	4472	+%211
# Kod satırı	220.986	450.952	+%104
KeePass			
-Ölçüt	Masaüstü sürümü	Mobil sürümü	Fark oranı (%)
# Dosya	194	362	+%86
# Sınıf	109	350	+%221
# Kod satırı	18.438	27.542	+%49



```
// Copyright 2003-2010 Christian d'Heureuse, Inventec Informatik AG, Zurich, Switzerland
// www.source-code.biz, www.inventec.ch/chdh
//
// This module is multi-licensed and may be used under the terms
// of any of the following licenses:
//
//  EPL, Eclipse Public License, http://www.eclipse.org/legal
//  LGPL, GNU Lesser General Public License, http://www.gnu.org/licenses/lgpl.html
//  AL, Apache License, http://www.apache.org/licenses
//  BSD, BSD License, http://www.opensource.org/licenses/bsd-license.php
//
// Please contact the author if you need another license.
// This module is provided "as is", without warranties of any kind.

package biz.source_code.base64Coder;
```

```
/*
 * This is the source code of Telegram for Android v. 3.x.x.
 * It is licensed under GNU GPL v. 2 or later.
 * You should have received a copy of the license in this archive (see LICENSE).
 *
 * Copyright Nikolai Kudashov, 2013-2016.
 */

package org.telegram.ui;
```

```
/*
 * Copyright 2009-2011 Brian Pellin.
 *
 * This file is part of KeePassDroid.
 *
 *  KeePassDroid is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  KeePassDroid is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with KeePassDroid.  If not, see <http://www.gnu.org/licenses/>.
 *
 */
package com.keepassdroid;
```

Şekil 15. Kaynak Kod Telif Hakkı Örneği

İncelenen iki uygulama çiftine ait toplam kaynak kodu boyutu ölçütleri için bulunan değerler ve masaüstü sürüm ile mobil sürüm arasındaki farklar Tablo 3'de verilmektedir. Telegram uygulama çifti için dosya sayısı, sınıf sayısı ve kod satırı sayısının mobil sürümde, masaüstü sürümden daha fazla olduğu görülmektedir. Benzer şekilde KeePass uygulama çifti için verilere bakıldığında, mobil sürümün masaüstü sürümden daha fazla dosya sayısına, sınıf sayısına ve kod satırı sayısına sahip olduğu görülmektedir.

Tablo 4. Üçüncü Parti Kaynak Kodu Boyut Ölçütleri ve Platforma Göre Farklılıkları

Telegram			
Ölçüt	Masaüstü sürümü	Mobil sürümü	Fark oranı (%)
# Dosya	7	1015	+%14,400
# Sınıf	0	202	-
# Kod satırı	1,826	215,966	+%11,720
KeePass			
Ölçüt	Masaüstü sürümü	Mobil sürümü	Fark oranı (%)
# Dosya	-	59	-
# Sınıf	-	35	-
# Kod satırı	-	6,121	-

KeePass ve Telegram uygulamalarının masaüstü sürümleri QT üçüncü parti kütüphanesini kullanmaktadır. QT ile Android platform kaynak kodları, platforma özel kaynak kodu olarak değerlendirildiği için, eş işleve sahiptir ve platforma ait bu kütüphanelerin kaynak kodları, Github üzerinden elde ettiğimiz uygulamaların kod tabanlarında yer almamaktadır. Bu nedenle Android sürümler için Android platformu kaynak kodları, masaüstü sürümler için QT kaynak kodu, kod boyutu incelemelerine dâhil edilmemiştir. Dolayısıyla başta amaçlandığı gibi üçüncü parti kaynak kodu boyutları elde edilememiştir.

Tablo 5. Projeye Özel Kaynak Kodu Boyut Ölçütleri ve Platforma Göre Farklılıkları

Telegram			
Ölçüt	Masaüstü sürümü	Mobil sürümü	Fark oranı (%)
# Dosya	505	836	+%65
# Sınıf	1,435	4,270	+%197
# Kod satırı	185,195	234,830	+%26
KeePass			
Ölçüt	Masaüstü sürümü	Mobil sürümü	Fark oranı (%)
# Dosya	194	204	+%5
# Sınıf	109	291	+%167
# Kod satırı	18,438	13,953	-%24

Tablo 5'te görüldüğü gibi uygulamaların mobil sürümleri, daha fazla sınıf ve kaynak kodu dosyası içermektedir. Telegram uygulamasının mobil sürümü için projeye özgü kaynak kod satırı, masaüstü sürümünden %25 daha fazladır. KeePass uygulama çiftine bakıldığında ise mobil sürümün masaüstü sürümden

%24 daha az kod satırı içerdiği görülmektedir. Dosya sayısı ve sınıf sayısı değerlerine bakıldığında ise her iki uygulama çifti için, mobil sürümlerin masaüstü sürümlerden daha büyük değerlere sahip olduğu görülmektedir.

4.3 AS3 Mobil ve Masaüstü Platformlar için Geliştirilmiş Uygulamalar, Farklı Türden Bağımlılıklar Açısından Nasıldır?

Bu soru cevaplanırken uygulamaların geliştirme dili, üçüncü parti kaynak kodu, projeye özel kaynak kodu ve platforma özel kaynak koduna olan bağımlılıkları hesaplanıp karşılaştırılmıştır.

Syer ve arkadaşları tarafından tanımlanan platform bağımlılık oranını ifade eden ölçüt, mobil ve masaüstü uygulamaların platform bağımlılıklarını ölçmek amacıyla bu çalışmaya uygun şekilde adapte edilmiş ve kullanılmıştır [4]. Buna göre bağımlılıklar, Bölüm 2.4'te tanımlandığı üzere; dil, platform, üçüncü parti kütüphane ve projeye olan bağımlılıklar olarak dört kategoriye ayrılmıştır.

Syer ve arkadaşları Platform Bağımlılık Oranı'nı (PBO), platform ve kullanıcı arayüzü bağımlılıklarının toplam bağımlılık sayısına oranı olarak tanımlamıştır (Denklem 1). Bu tez çalışmasında, platform bağımlılığının araştırılması hedeflendiği için bu iki bağımlılığın ayrı olarak incelenmesine gerek görülmemiştir. Bu nedenle platform bağımlılığı ve kullanıcı arayüz bağımlılığı bir arada değerlendirilip platform bağımlılığı olarak hesaplanmıştır. Platform bağımlılık oranı ise Denklem 2'de verildiği gibi tanımlanmıştır:

$$\mathrm{PBO} = \sum_{i=0}^{n} \left(\frac{P_i}{D_i + P_i + 3rd_i + PR_i} \right) / n \qquad (2)$$

P: Platform sınıflarına bağımlılık sayısı
D: Dil sınıflarına bağımlılık sayısı
3rd: Üçüncü parti kütüphane sınıflarına bağımlılık sayısı
PR: Proje sınıflarına bağımlılık sayısı
n: Toplam sınıf sayısı

Bağımlılıkları çıkartırken Understand aracından yararlanılmıştır. Understand, bağımlılıkları aşağıda belirtilen türlerde çıkartmaktadır;

- Calls – başka bir sınıftaki yönteme (İng. Method) çağrı yapmak
- Casts – başka bir sınıfta tanımlanmış bir nesne türüne çevirmek
- Creates – başka bir sınıfta tanımlanmış bir nesne türünden nesne yaratmak
- Extends – başka bir sınıfa dâhil etmek

- Implements – başka bir ara yüzü uygulamak
- Sets – başka bir sınıfta tanımlı bir değişken ya da nesneye atama yapmak
- Typeds – başka bir sınıfta tanımlanmış bir nesne türünü kullanmak
- Uses – başka bir sınıfta tanımlanmış bir nesyeyi ya da değişkeni kullanmak**.**

Denklem 2 için bağımlılıklar hesaplanırken bir sınıftan diğer bir sınıfa olan bir bağımlık, bağımlılık türünden ve sayısından bağımsız olarak tek sayılmıştır. Amaç bir sınıfın hangi sınıflar ile bağımlılığı olduğunu ortaya koymaktır. Understand tarafından çıkartılan bağımlılıklar, bağımlı olunan sınıflar incelenerek; platform, dil, üçüncü parti kütüphane ve projeye ait bağımlılıklar olarak ayrıştırılmıştır. Bu aşamada Understand aracının, incelenen yazılım projesinin kaynak koduna dahil olmayan sınıflar için bağımlılıkları hesaplayamadığı fark edilmiştir. Bu nedenle her sınıfın kaynak kodu incelenerek eksik bağımlılıklar çıkartılmıştır. Bu işlem yapılırken sınıfın tanımı, yöntemleri ve sınıfa dahil edilen (import) diğer sınıflar taranarak sayılmış ve Understand aracı tarafından çıkartılan bağımlılıklara eklenmiştir. Telegram ve KeePass uygulamalarının masaüstü sürümleri C++ ile geliştirilmiştir. Masaüstü sürümlere ait, Understand tarafından çıkartılan bağımlılıklarda, sınıfların tanımlandığı dosya adı/yolu belirtilmediği için her sınıf kaynak kodu tabanında aratılarak bulunmaya çalışılmış, bu da bağımlılıklar çıkartılırken harcanan eforu artırmıştır. Toplam 1435 sınıfı bulunan Telegram uygulamasının masaüstü sürümü için bağımlılıklar çıkartılmıştır.

Tablo 6. Uygulama Çiftlerinin Platform Bağımlılık Oranı

Uygulama	PBO (sıfırlar dâhil ortalama)	PBO (sıfırlar hariç ortalama)
Telegram Mobil	0,45420	0,63860
Telegram Masaüstü	0,30830	0,46210
KeePass Mobil	0,33333	0,58083
KeePass Masaüstü	0,42333	0,45991

PBO, her bir sınıf için, dört kategoride çıkartılan bağımlılık sayıları ile Denklem 2 hesaplanarak elde edilmiştir. Bazı sınıfların platform koduna hiç bağımlılığı olmaması nedeniyle PBO değeri sıfır olarak hesaplanmıştır. Her uygulama için PBO değerlerinin ortalaması, sıfırlar dâhil ve sıfırlar hariç olarak hesaplanmıştır. Tablo 6'te verilmiş olan sıfırlar dâhil ortalamalara bakıldığında Telegram

uygulaması için mobil sürümün ortalama PBO değerinin, masaüstü sürümün PBO değerinden daha yüksek olduğu görülmektedir. KeePass uygulaması için ise masaüstü sürümün ortalama PBO değerinin mobil sürümün ortalama PBO değerinden yüksek olduğu görülmektedir. Platforma bağımlılığı olmayan sınıfların hariç tutulduğu ortalama PBO değerlerine bakıldığında ise her iki uygulama çifti için mobil sürümlerin ortalama PBO değerleri, masaüstü sürümlerin ortalama PBO değerlerinden yüksektir.

4.4 AS4 Bu İki Platform İçin Geliştirilen Uygulamalar Arasında Benzerlik Gözetilebilir mi?

Bundan önceki araştırma sorularında uygulama çiftlerinin nesneye yönelik tasarım ölçütleri, kaynak kod boyutu ve bağımlılıklar açısından nasıl olduğu sorulmuş ve her bir araştırma sorusu için veriler elde edilmiştir. Bu araştırma sorusu ile bu veriler ve verilerin elde edilme koşulları değerlendirilerek mobil ve masaüstü platformların, uygulamalar üzerinde bir etkisi olup olmadığını ve platformlar arası benzerlik veya farklılıkların gözetilip gözetilemeyeceğini anlamak amaçlanmıştır.

Bir uygulamanın kaynak kodunu etkileyen etmenler oldukça çeşitlidir. Geliştirme dilinin yetenekleri, platformun geliştiricilere sunduğu olanaklar, geliştiricilerin deneyimi ve sayısı, uygulamanın ne kadar süredir geliştirildiği bu etmenlere örnek olarak verilebilir. Etmenlerin tamamı hakkında bilgi sahibi olmak güçtür. Öte yandan bu etmenlerdeki farklılıkların uygulamaları ne ölçüde etkilediğinin tespiti de oldukça güçtür.

Bu tez kapsamında incelenen uygulamalar seçilirken denk özelliklere sahip olmalarına dikkat edilmiştir. Aynı dille geliştirilmiş uygulama çiftlerine ulaşılamadığı için, nesneye yönelik dillerle geliştirilmiş uygulamaların seçilmesine özen gösterilmiştir. Arifoğlu ve Doğru, işlev noktalarından farklı yazılım geliştirme platformlarına (ve dile) göre satır sayısı kestirimi için bir ortalama değer önermiştir. Örneğin, bu ortalama değer C geliştirme dili için 90 iken COBOL ve FORTRAN için 100'dır. Nesne-kökenli diller ise bir arada değerlendirilmiş ve onlar için verilen değer 30'dur. Öte yandan, Larry Putnam tarafından 1978'de kurulan QSM (Quantitative Software Management), işlev noktalarından yararlanarak geliştirilmiş 2192 proje ile 32 farklı geliştirme dili için çevrim faktörünü tanımlamaktadır [27]. QSM'in bu çevrim faktörlerinin yer aldığı İşlev Noktaları Tablosu'nda dillere ait minimum, maksimum, ortalama ve medyan çevrim faktörlerinin değerleri

verilmektedir. C++ ve Java için ortalama ve medyan çevrim faktörü değerleri çift olarak sırasıyla 50-53 ve 50-53'tür. Bu bilgiler ışığında, kaynak kod boyutu incelemesinde, uygulamaların farklı dillerle geliştirilmiş olmasının sonuçlara bir etkisi olmadığı yorumu yapılabilir.

Tablo 7. Uygulamaların Sürüm ve Geliştirici Sayıları

Uygulama	Sürüm Sayısı	Toplam Geliştirici Sayısı	Son Sürüm Bilgisi
KeePass Masaüstü	12	29	v 2.0.3 (26/11/2017)
KeePass Mobil	156	34	v 2.3.4 (25/06/2018)
Telegram Masaüstü	324	75	v 1.3.12 (05/08/2018)
Telegram Mobil	0	11	v.0 (30/07/2018)

Tablo 7'da bu tez çalışması kapsamında incelenen uygulamaların sürüm sayıları, geliştirici/katkı sağlayıcı (İng. contributor) sayıları ve son sürüm adı ve tarihi görülmektedir. KeePass mobil uygulamasının 156 sürümü ve 34 katkı sağlayıcısı bulunmaktadır. Masaüstü uygulamanın ise 12 sürümü ve 29 katkı sağlayıcısı bulunmaktadır. Telegram için masaüstü uygulamanın 324 sürümü ve 75 katkı sağlayıcısı olduğu görülmektedir. Mobil uygulama ise tek sürüm olarak yayınlanmakta ve uygulamanın 11 katkı sağlayıcısı bulunmaktadır. Son güncelleme tarihi Tablo 7'da görülmektedir. Her ne kadar uygulamalarda özellik denkliği ve nesneye yönelik dillerle geliştirilmiş olması gibi kriterlere dikkat edildiyse de sürüm sayıları ve geliştirici sayısı gibi farklılıkların kaynak kodu büyük ölçüde etkilediği düşünülmektedir. Açık kaynak kodlu uygulama geliştirme platformlarında, hata düzeltme ve dil eklentisi gibi minör değişiklikler de katkı sağlayıcılar tarafından yapılabilmektedir. Bu durum açık kaynak kodlu uygulamalara erişilen platformlarda yer alan katkı sağlayıcı sayısını etkilemekte ve gerçek geliştirici sayısını öğrenmeye engel olmaktadır. Dolayısıyla, geliştirici sayısı ve geliştirici tecrübesinin bilinmemesi, kaynak koda dayanarak bu iki platformu karşılaştırmak için bir taban oluşturmaya imkân vermemektedir. Bu açıdan bu iki platform için eldeki verilere dayanarak benzerlik gözetilemeyeceği sonucuna varılabilir.

Platformlar geliştiricilere gerek yazılımsal gerekse donanımsal olarak farklı olanaklar sunar. Bir platform için geliştirilmiş üçüncü parti bir kütüphane başka bir platform için erişilebilir olmayabilir. Bu noktada geliştirici, bu ihtiyacı kendi

geliştirmek durumunda kaldığı bir kod ile giderebilir. Aynı zamanda, ihtiyaca uygun üçüncü parti kütüphanelerin olması durumunda dahi, bu kütüphanelerin kullanımı geliştiricinin tasarrufundadır. Üçüncü parti kütüphanelerin daha çok tercih edilmiş olması, platforma olan güvenin az olması veya platformun bu yeteneği karşılayamamasından kaynaklanabildiği gibi geliştirici tercihinden de kaynaklanıyor olabilir. Bu açıdan, sadece kaynak koddaki projeye özel geliştirilmiş kodun boyutu ve üçüncü parti kodun boyutuna bakmak, platformların üçüncü parti koda olan güvenleri veya bağımlılıkları açısından çıkarım yapmak için yetersizdir. Öte yandan tek başına projeye özel geliştirilmiş kaynak kodun boyutu, kaynak kod geliştirme çabasının bir göstergesi olması bakımından daha anlamlıdır. Fakat daha önce bahsedildiği gibi, geliştirici tecrübesi ve sayısı ile geliştirme dilinin etkilerinden ötürü, mobil ve masaüstü için farklı dillerle geliştirilmiş uygulamalar açısından platformlar arasında benzerlik gözetmek için, kaynak kod boyutu tek başına yetersiz kalmaktadır.

Nesneye yönelik ölçütler ile kaynak kodu analiz edilerek yazılımın tasarımı hakkında fikir sağlanabilir. Bu tez çalışmasında tercih edilen C&K ölçüt kümesi için elde edilen verilere bakıldığında, RFC ve WMC ölçütleri için hem dağılımda hem de ortalama değerlerde benzerlikler gözlenmiştir; DIT ve CBO ölçütleri için ise ortalama değerlerde benzerlikler gözlenmiştir. C&K ölçüt kümesi ile yazılımın çok-biçimlilik, kalıtım, soyutlama gibi özelliklerine bakarak bakım yapılabilirlik, anlaşılabilirlik ve yeniden kullanılabilirlik gibi yazılım kalite özellikleri hakkında yorum yapılabilse de bu ölçütler, bir yazılımın bu gibi özelliklerini tek başına değerlendirmek için eksiktirler. Örneğin, sınıfların daha çok uygulamaya özgü olup yeniden kullanılabilirliği olumsuz etkilediği söylenen sınıf başına ağırlıklı metot sayısı (WMC) ölçütü için yapılan ölçümlerde, her iki uygulama çifti için masaüstü sürümlerin daha yüksek değerlere sahip olduğu gözlemlenmiştir. Bu noktada, her iki uygulama çifti için masaüstü sürümlerin WMC değerinin yüksek olması bulgusundan yola çıkarak, masaüstü uygulamalarda sınıfların daha çok uygulamaya özgü olduğu ve yeniden kullanılabilirliğin düşük olduğu çıkarımı yapılabilse de tek başına genelleştirilmek için yetersizdir. Benzerliklerin gözlendiği C&K ölçütleri, kaynak kodu etkileyen diğer etmenlerle ele alınarak bir arada değerlendirildiklerinde daha güçlü yorumlar yapmaya olanak sağlayabilirler.

Özet olarak, her araştırma sorusu kapsamında mobil ve masaüstü uygulamalar arasında, platform bakış açısıyla karşılaştırma yapılabileceğini işaret eden benzerlikler gözlenmiştir. Fakat yukarıda bahsedilen sonuçların geçerliliğini etkileyebilecek hususlar göz önünde bulundurulduğunda, farklı dillerle geliştirilmiş denk özellikteki mobil ve masaüstü uygulamaların nesneye yönelik tasarım, kaynak kod boyutu ve bağımlılıklar bakımından bu keşifsel araştırma kapsamında incelendiği şekliyle karşılaştırılması, bu iki platformun benzerliğini tartışarak genelleştirilebilir çıkarımlar yapmak için elverişli bulunmamıştır.

5. TARTIŞMA

Bu bölümde bu tez çalışması kapsamında elde edilen bulguların özeti, sonuçların geçerliliğine yönelik tehditler ve öneriler ele alınmaktadır.

5.1 Bulguların Özeti

Bu tez çalışması kapsamında, dört ana araştırma sorusu belirlenmiştir.

Birinci araştırma sorusu (AS1) ile uygulamaların masaüstü ve mobil sürümlerinin yazılım tasarım özelliklerinin nasıl olduğu, yazılım tasarım ölçütlerinden yararlanarak incelenmiştir. C&K ölçüt seti kullanılarak yapılan karşılaştırmada her ölçüt için ölçümler yapılmış ve veriler ortalama değerler ve kutu grafiği üzerinden incelenmiştir. Her ölçüt için elde edilen sonuçlar aşağıda özetlenmiştir:

- Sınıfın tetiklediği metot sayısı (RFC) için verilerin %95'inin yer aldığı güven aralığına bakıldığında ve her iki uygulama çifti bir arada değerlendirildiğinde mobil sürümlerin, masaüstü sürümlere göre daha yüksek RFC değerlerine sahip olduğu söylenebilir.
- Metotlardaki uyum eksikliği (LCOM) ölçütü için yapılan ölçümlerde, Telegram'ın masaüstü ve mobil sürümünün ortalama LCOM değerleri ve değerlerin dağılımı, birbirinden önemli ölçüde farklılık göstermemektedir. Az farkla mobil sürümün ortalaması daha yüksek ve dağılım aralığı daha geniştir. KeePass uygulama çiftine bakıldığında ise masaüstü sürümün LCOM değerlerinin hem dağılım hem de ortalama değer bakımından daha yüksek olduğu görülmektedir.
- Sınıfın ağırlıklı metot sayısı (WMC) için yapılan ölçümlerde, KeePass uygulaması için iki sürümün ortalama WMC değerleri arasındaki fark daha fazla olsa da her iki uygulama çifti için, masaüstü sürümlerin daha yüksek WMC değerlerine sahip olduğu gözlemlenmiştir. WMC değerinin yüksek olması sınıfların daha çok uygulamaya özgü olduğunu ve yeniden kullanılabilirliğin bundan olumsuz etkilendiğini gösterir. Bu bilgi ışığında WMC değerlerinin mobil uygulamalar için daha yüksek olması, mobil uygulamaların daha çok platforma özgü olduğuna işaret edebilir.
- Her iki uygulama çifti için de masaüstü sürümlerin DIT değerinin, mobil sürümlere ait değerlerden daha yüksek olduğu saptanmıştır. DIT değerinin yüksek olmasının, kalıtımdan da gelecek olan metotlar düşünüldüğünde

sınıfın karmaşıklığını artırdığı, buna ek olarak bakım ve değişiklik maliyetlerinin daha fazla olması ihtimalini de artırdığı düşünülmektedir. Masaüstü uygulamalarda performans kaygısının olmaması, masaüstü uygulamalarda DIT değerinin daha yüksek olmasına ve dolayısıyla karmaşıklığın da daha yüksek olmasına sebep olmuş olabilir.

- Nesne sınıfları arasında bağımlılık (CBO) için uygulama çiftlerinde ortalama değerlere ve dağılıma bakıldığında, her iki uygulama çifti için de mobil sürümlerin sınıflar arası bağımlılığının daha yüksek olduğu görülmektedir. Nesne sınıfları arasındaki aşırı bağlanmanın, modüler tasarım için zararlı olduğu ve yeniden kullanılabilirliği olumsuz etkilediği düşünülmektedir [13]. Bu açıdan mobil uygulamaların, daha uygulamaya özgü olmaları muhtemeldir.
- Alt sınıf sayısı (NOC), bir temel sınıftan türetilen anlık çocuk sınıflarının sayısına eşittir. Her iki uygulama çifti için de dağılımların net bir formu olmamakla birlikte ortalama değerlere bakıldığında, KeePass uygulama çifti için mobil sürümün, Telegram uygulama çifti için ise masaüstü sürümün daha yüksek değere sahip olduğu gözlenmiştir.

Kaynak kod boyutu ölçütlerinin, bir yazılım sisteminin karmaşıklığı ile önemli ölçüde ilişkili olduğu kanıtlanmıştır [22]. İkinci araştırma sorunu (AS2) ile uygulamaların mobil ve masaüstü sürümlerinin kaynak kodu boyutları incelenmiştir. Dosya sayısı, sınıf sayısı ve kod satır sayısı (LOC) ölçütleri için yapılan ölçümler sonucunda, uygulamaların mobil sürümlerinin boyutunun daha büyük olduğu bulunmuştur. Kaynak kod boyutu ve geliştirme çabası arasındaki ilişki göz önünde bulundurulduğunda, bir uygulamanın mobil sürümünü geliştirirken daha çok kaynak ve zaman gerekeceği düşünülebilir. Fakat uygulamaların mobil ve masaüstü sürümlerine ait geliştirici sayısı ve tecrübesinin bilinmemesi, net bir çıkarım yapmayı engellemektedir.

Üçüncü araştırma sorusu (AS3) ile uygulamaların farklı platformlar için geliştirilmiş sürümlerinin farklı türden bağımlılıklar açısından nasıl olduğu ve platform bağımlılık oranları araştırılmıştır. Geliştirme dili, üçüncü parti kaynak kodu, projeye özel kaynak kodu ve platforma özel kaynak koduna olan bağımlılıklar hesaplanarak platform bağımlılığının toplam bağımlılığa oranı tespit edilmiştir. Platforma bağımlı olan sınıfların ortalamasına bakıldığında her iki uygulama çifti

için, mobil sürümlerinin platform bağımlılık oranı daha yüksek çıkmıştır. Buradan yola çıkarak mobil sürümlerin üzerinde çalıştıkları platforma daha bağımlı olduğu algısı oluşmaktadır. Mobil ve masaüstü sürümler için farklı olan geliştirme dillerinin, sahip oldukları yetenekler açısından platform bağımlılığı üzerine etkisi olma ihtimali nedeniyle, bu sonuçlar farklı ölçütler ve yöntemler ile platform bağımlılığının araştırılması ile desteklenmelidir.

Dördüncü araştırma sorusu (AS4) ile ilk üç araştırma sorusu kapsamında elde edilen veriler ve bu verileri etkileyen etmenler değerlendirilerek mobil ve masaüstü platformların uygulamalar üzerinde bir etkisi olup olmadığı yorumlanmış ve platformlar arasında benzerlik veya farklılıkların gözetilip gözetilemeyeceği araştırılmıştır. Birinci araştırma sorusu kapsamında, C&K ölçütleri için elde edilen sonuçlarda, RFC ve WMC ölçütleri için hem dağılımda hem de ortalama değerlerde benzerlikler gözlenmiştir; DIT ve CBO ölçütleri için ise ortalama değerlerde benzerlikler gözlenmiştir. İkinci araştırma sorusunda, uygulamaların mobil sürümlerine ait kaynak kod boyutunun, mobil sürümlerde daha büyük olduğu bulunmuştur. Platform bağımlılığının araştırıldığı üçüncü araştırma sorusunda, mobil uygulamaların platforma olan bağımlılıklarının daha yüksek olduğu gözlenmiştir. Bu bulgular, mobil ve masaüstü uygulamaların platform açısından benzerlik ve farklılık bakış açısıyla ele alınabileceğini göstermektedir. Denk özelliklerde mobil ve masaüstü uygulama çiftlerinin incelenmiş olması ve uygulamaların nesneye yönelik dillerle geliştirilmiş olmaları bir avantajdır. Bununla birlikte geliştirici tercihleri, tecrübesi ve sayısı gibi kaynak kodunu etkileyen etmenlerin çeşitliliği, açık kaynak kodlu uygulamaların incelenmiş olması, bu uygulamalara erişim koşulları ve sonuçları etkileyebilecek etmenler hakkında bilgi sahibi olunamaması gibi nedenlerle, mobil ve masaüstü platformlar arasında bu araştırma bağlamında benzerlik ilişkisi kurulamamıştır.

5.2 Geçerliliğe Yönelik Tehditler

Bu bölümde, standart bir kontrol listesi temel alınarak [28] hazırlanmış, bu tez çalışmasına sınır teşkil edebilecek geçerlilik tehditlerinden ve bunların nasıl azaltıldığından bahsedilmektedir. Dört tip geçerliliğe tehdit dikkate alınmıştır.

5.2.1 İçsel Geçerlilik

Bu geçerlilik, bir çalışmanın elde edilen verilere dayalı nedensel bir sonuç garanti etmesi ile ilgilidir. AS2 için, incelenen uygulamalarda kullanılmış, üçüncü parti

kütüphanelere ait kaynak kodların belirlenmesi elle yapılmıştır; dolayısıyla insan kaynaklı hatalara sebep olunmuş olabilir. Ayrıca, uygulamalarda kullanılmış fakat kaynak kodu kod tabanında yer almayan üçüncü parti kütüphaneleri olabilir. Bunlar sonuçları etkileyebilecek faktörlerdir. Uygulamaların kaynak kodlarını değerlendirmede nesneye yönelik diller için önerilmiş olan bir ölçüt kümesi tercih edilmiştir. Böylelikle, mobil ve masaüstü uygulamaların farklı programlama diliyle geliştirilmiş olmasından kaynaklanabilecek tehdidi azaltmak hedeflenmiştir. Bölüm 4.4'te belirtildiği üzere, kaynak kod boyutu incelemesinde, mobil uygulamaların Java, masaüstü uygulamaların ise C++ ile geliştirilmiş olması bir tehdit oluşturmamaktadır. Ölçütler için gerekli değerler ve çizgeler, statik kod analiz aracı olan Understand [12] aracılığı ile elde edilmiştir. Uygulamaların denk özelliklerde mobil ve masaüstü sürümler olarak seçilmesinin, içsel geçerliliği arttırdığı düşünülmektedir. Bununla birlikte, uygulama çiftlerinin (Telegram ve KeePass) farklı işlevsel özelliklere sahip olması, bir iç tehdit olarak düşünülebilir; ancak uygulamalar birbiri arasında karşılaştırılmadığı için, bu tehdit göz ardı edilebilir.

5.2.2 Yapısal Geçerlilik

Bu geçerlilik, çalışmanın nesnelerinin çalışmanın arkasındaki teoriyi temsil etme ölçüsü ile ilgilidir. Bu tez çalışmasında kullanılmış olan yaklaşım, mobil uygulamalar üzerine kod tabanlı incelemelerin yapıldığı literatürdeki çalışmalara [4][6] benzemektedir. Çalışmalarında nesneye yönelik ölçütleri kullananların, çoğunlukla tercih etmesinden [21] yola çıkılarak AS1 için yazılım tasarım özellikleri değerlendirilirken C&K ölçüt kümesi kullanılmıştır. AS3'de uygulamaların platform bağımlılığı değerlendirilirken Syer ve arkadaşlarının önerdiği ölçüt [4], bu tez çalışmasına uygun olacak şekilde adapte edilip kullanılmıştır. AS2'de kullanılan kod boyutu ölçütleri seçilirken de Syer ve arkadaşlarının çalışmasından [4] yararlanılmıştır.

5.2.3 Sonuç Geçerliliği

Bir çalışmanın sonuç geçerliliği, onun sonuçlarının titiz ve tekrarlanabilir bir yöntem ile tekrar ulaşılabilir olup olmadığı ile ilgilidir. Bu çalışmada, uygulamaların kaynak kodu boyutu özellikleri Bölüm 3'te bahsedilen yöntem ile niceliksel olarak ölçülmüştür. C&K ölçütleri için elde edilen ölçümler Understand aracı ile elde edilmiştir ve tekrar edilebilir sonuçlardır. AS3 ile elde edilen sonuçlar, Understand aracı ile elde edilen sonuçlara, kaynak kodu dosyalarının tek tek taranmasıyla elde

edilen sonuçların birleştirilmesiyle elde edilmiştir. Çalışma titizlikle yürütülmüştür ve var ise kişisel hataların sonuçlara etkisinin az olduğu düşünülmektedir. Sonuçlar uygulamaların kaynak kodlarına erişildiği tarihteki sürümleri için geçerlidir. Bunun yanısıra, kaynak kodu ve yazılım tasarımı üzerinde etkili olabilecek etmenler olan geliştirici deneyimi ve sayısı, uygulamanın sürüm sayısı ve geliştirme süresi hakkında detaylı bilginin erişilmez olması, tez çalışmasının sonuç geçerliliği için bir tehdittir.

5.2.4 Dış Geçerlik

Dış geçerlilik bir çalışmanın sonuçlarının genelleştirebilir olma ölçüsü ile ilgilidir. Çalışılan uygulama çiftleri, mobil ve masaüstü platformları için farklı dillerle geliştirilmiş denk özellikteki açık kaynak kodlu uygulamaların küçük bir alt kümesini temsil etmektedir. Bu çalışma gelecek çalışmalar için bir öncü olarak amaçlanmış, dolayısıyla çalışmada dış geçerlilik hedeflenmemiştir. Bu sebeple elde edilen sonuçlar, diğer uygulamalar veya platformlar için genelleştirilemez.

5.3 Öneriler ve Gelecek Çalışmalar

Bulgular ve AS4 çerçevesinde belirtilen yorumlar göz önünde bulundurulduğunda, gelecekte yapılacak çalışmalar ve araştırmalar için dikkat edilmesi gereken hususlar ve öneriler aşağıda belirtilmiştir:

- Geliştirme dilinin aynı olmasına dikkat edilmelidir.
- Geliştirici tecrübesi ve sayısının denk olması, sağlanamıyor ise bu bilgilere sahip olunması gereklidir.
- Üçüncü parti kütüphane kullanımı hakkında detaylı bilgiye sahip olunması ve çalışmanın amacına göre mümkün ise kontrol edilmesi sağlanmalıdır.

Bu keşifsel araştırma sırasında karşılaşılan zorluklar ve edinilen tecrübelerden yola çıkılarak sunulan önerilerin uygulanabileceği en uygun araştırma ortamı, küçük çaplı projelerin belirli ekiplerle, kontrollü bir ortamda geliştirilmesi ile elde edilebilir.

Gelecek çalışmalarda, benzerliklerin gözlendiği C&K ölçütlerinden yola çıkarak mobil ve masaüstü platformun tasarım özelliklerine etkisi üzerine, yeni araştırma soruları yöneltilebilir. Bu alanda, ilişkili ölçütlerin birlikte kullanımı ile bir veya birkaç yazılım özelliğinin ölçülmesine odaklanarak daha güçlü sonuçlar elde edilebilir. Ayrıca, mobil ve masaüstü platformların tasarım üzerindeki olası etkilerinin Model

Güdümlü Yazılım Geliştirme bağlamında ele alınarak incelenmesi, modellerdeki öğeler ve modeller arasındaki dönüşümler açısından karşılaştırmada daha somut bir temel sağlayabilir. Mobil uygulamalarda daha yüksek olması beklenen platform bağımlılığı, daha fazla sayıda denk özellikteki uygulamalar üzerinde sebep ve etkileri çerçevesinde derinlemesine araştırılabilir. Sadece ölçüt bakış açısıyla ele alınan bu çalışmanın devamında, uygulamalar mimari bakış açısıyla incelenebilir. Uygulamaların mobil ve masaüstü sürümlerinin kaynak kod dosyalarında sınıf ve metot isimlerine bakılarak benzerlikler incelenebilir.

6. SONUÇ

Aynı uygulamayı birden fazla platform için geliştirmek her platformun kendine özgü bir geliştirme ortamı, geliştirme dili ve geliştirme teknolojisinin olmasından kaynaklı olarak gerektirdiği kaynak, zaman ve çaba açısından zorlu bir iştir. Mobil ve masaüstü platformların karşılaştırıldığı ve bu platformlar için uygulama geliştirmeyi konu alan, geliştirici bakış açısıyla ele alınmış çalışmaların ve bu alandaki birikimin azlığı nedeniyle bu tez çalışmasında, keşifsel bir araştırma yapılmıştır. Mobil ve masaüstü platformlar için geliştirilmiş uygulamalar tasarım ölçütleri, kaynak kod boyutu ve platform bağımlılığı açısından incelenmiştir. Bulgulara ve araştırma aşamasında karşılaşılan zorluk ve tecrübelere dayanarak mobil ve masaüstü platformlar için geliştirilmiş uygulamalar arasında benzerlik gözetilip gözetilemeyeceği araştırılmıştır. Keşifsel yöntemle elde edilen sonuçların, mobil ve masaüstü platformlar için benzer teknolojiler ile uygulama geliştirecek geliştiricilere fikir vermesi ve gelecekte bu alanda çalışmalar yapacak araştırmacılara bir bakış açısı sağlaması ve yeni araştırma alanları belirlemeye yaraması beklenebilir.

Birinci araştırma sorusu (AS1) kapsamında tasarım değerlendirmesinde, C&K ölçüt kümesinden yararlanılmıştır. DIT ve WMC ölçütleri için elde edilen sonuçlarda, masaüstü sürümlerin mobil sürümlerden daha yüksek değerlere sahip olduğu, RFC ve CBO ölçütü için elde edilen sonuçlarda ise mobil sürümlerin değerlerinin daha yüksek olduğu görülmüştür. DIT ve WMC'nin yüksek olması masaüstü uygulamaların daha karmaşık olabileceği ve bakım maliyetlerinin daha yüksek olabileceği anlamı taşıyabilir. Bu alanda, kaynak kodu etkileyebilecek diğer etmenler de göz önünde bulundurulup daha detaylı çalışmalar yapılmalıdır.

İkinci araştırma sorusu (AS2) kapsamında, uygulama çiftleri kaynak kod boyutu açısından incelenmiş ve her iki uygulama çifti için, mobil sürümlerin kod boyutunun daha büyük olduğu görülmüştür. Kaynak kod boyutu ve geliştirme çabası arasındaki ilişki göz önünde bulundurulduğunda, bir uygulamanın mobil sürümünü geliştirirken daha çok kaynak ve zaman gerektirdiği düşünülebilir. Fakat kaynak kodu etkileyebilen diğer etmenler düşünüldüğünde bu sonuçlar geliştirme çabası ile ilişkilendirilmemiştir.

Üçüncü araştırma sorusu (AS3) ile dört farklı türden bağımlılıklar gözetilerek uygulamaların üzerinde çalıştığı platforma olan bağımlılıkları hesaplanmış ve

incelenmiştir. Platform bağımlılığı uygulamanın başka platformlara taşınmasını da güçleştirebilen bir etmendir. Uygulamaların mobil sürümlerinin platform bağımlılık oranı daha yüksek olarak gözlenmiştir. Bu bulgu, mobil uygulamaların üzerinde çalıştığı platforma daha bağımlı olduğuna işaret edebilir. Platform bağımlılığının yüksek olması, platformun işletim sisteminde yapılacak güncelleme ve değişikliklerin uygulamayı daha çok etkileyebileceği anlamı da taşımaktadır. Mobil işletim sistemlerinin sık güncelleme aldığı göz önünde bulundurulursa, mobil uygulamaların değişiklik maliyetinin ve platforma bağlı hatalarının yüksek olması mümkündür.

Son araştırma sorusu (AS4) ile mobil ve masaüstü platformların uygulamalar üzerinde bir etkisi olup olmadığı yorumlanarak platformlar arası benzerlik veya farklılıklar gözetilebilir mi sorusunun cevaplanması amaçlanmıştır. İlk üç araştırma sorusu kapsamında elde edilen veriler ve bu verileri etkileyen etmenler değerlendirilmiş ve geliştirici tercihleri, tecrübesi ve sayısı gibi kaynak kodunu etkileyen etmenlerin çeşitliliği ve açık kaynak kodlu uygulamaların incelenmiş olması nedeniyle, bu bulguların benzerlik ilişkisi kurmak için zayıf olduğu düşünülmüştür. Uygulamalara erişim koşulları dolayısıyla sonuçları etkileyebilecek etmenler hakkında bilgi sahibi olunamaması nedeniyle, mobil ve masaüstü platformlar arasında bu araştırma bağlamında doğrudan benzerlik ilişkisi kurmak güçtür. Diğer yandan, kaynak kod boyutunda ve tasarım ölçütlerinin değerlerinde gözlemlenen benzerlikler, sonuçları etkileyen etmenlerin daha kontrollü değerlendirilebildiği koşullarda, mobil ve masaüstü platformlar için daha güçlü bulguların elde edilebileceğine ve daha net çerçevelerde benzerlik ilişkisi araştırılabileceğine işaret etmektedir.

Yazılım mühendisliğinin çalışma doğası gereği, araştırmalarda birçok değişken kontrol edilememektedir ve her etmen hakkında bilgi toplamak ve bunların etkisi ölçmek güçtür. Bu keşifsel araştırma sırasında karşılaşılan zorluklar ve edinilen tecrübelerden yola çıkılarak, platformların yazılım üzerindeki etkilerini daha sağlam bir temelde belirlemek için; küçük çaplı projelerin belirli ekiplerle, kontrollü bir ortamda geliştirilmesi veya endüstri ve akademi iş birliği ile yürütülecek araştırma çalışmalarının planlanarak yürütülmesi düşünülebilir.

KAYNAKLAR

1. Number of mobile app downloads worldwide in 2016, 2017 and 2021, https://www.statista.com/statistics/271644/worldwide-free-and-paid-mobile-app-store-downloads/ (Şubat, **2018**).
2. Liu J., Yu J., Research on development of android applications, *Fourth International Conference on Intelligent Networks and Intelligent Systems*, 1-3 Kasım, Kunming, China, **2011.**
3. Schmidt D.C., Model-Driven Engineering, *IEEE Computer*, 39, 25–31, **2006**
4. Syer M.D., Adams B., Zou Y., Hassan A.E., Exploring the development of micro-apps: A case study on the blackberry and android platforms, *In Proceedings of 11th IEEE International Working Conference on Source Code Analysis and Manipulation*, 55–64, **2011**.
5. Syer M.D., Nagappan M., Hassan A.E., Adams B., Revisiting Prior Empirical Findings For Mobile Apps: An Empirical Case Study on the 15 Most Popular Open-Source Android Apps, *Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research*, 283–297, **2013**.
6. Syer M.D., Nagappan M., Adams B., Hassan A.E., Studying the relationship between source code quality and mobile platform dependence, *Journal of Software Quality*, 23, 485–508, **2015**.
7. Easterbrook S., Singer J., Storey M.A., Damian D., Selecting Empirical Methods for Software Engineering Research, *Guide to Advanced Empirical Software Engineering*, Springer, London, 285–311, **2008**.
8. Oivo M, Kuvaja P., Pulli P., Similä J., Software Engineering Research Strategy: Combining Experimental and Explorative Research (EER), *Proc. of International Conference of Product Focused Software Process Improvement*, 5-8 April, Kausai Science City, Japan, 302–317, **2004**.
9. Chidamber S.R., Kemerer C.F., A Metrics Suite for Object Oriented Design, *IEEE Transactions on Software Engineering*, 20, 476–493, **1994**.
10. Abreu F.B., Carapuça R., Object-Oriented Software Engineering: Measuring and Controlling the Development Process, *4th International Conference of Software Quallity*, 4, 3–5, **1994**.
11. Erdemir U., Tekin U., Buzluca F., Nesneye Dayalı Yazılım Metrikleri ve Yazılım Kalitesi, *Yazılım Kalitesi ve Yazılım Geliştirme Araçları Sempozyumu*, 9-10 Ekim, İstanbul, **2008**.
12. Lake A., Cook C., Use of factor analysis to develop OOP software complexity metrics, *6th Annual Oregon Workshop on Software Metrics*, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.56.7418&rep=rep1&type=pdf, **1994**.
13. Cankurtaran E., Çilden E., Tarhan A., Bileşen Tabanlı ve Ürün Hattı Yazılım Geliştirme Yaklaşımlarında Yeniden Kullanılabilirlik Metrikleri Yeniden Kullanım ve Yeniden Kullanılabilirlik, *XI. Ulusal Yazılım Mühendisliği Sempozyumu*, Alanya (CEUR Proceedings, sf. 529-540), **2017**.
14. Platform Dependent Definition by PC Magazine, https://www.pcmag.com/encyclopedia/term/49364/platform-dependent (Temmuz, **2018**).
15. Singh R., An Overview of Android Operating System and Its Security Features, *Rajinder Singh Int. Journal of Engineering Research and Applications,* 4, 519–521, **2014**.
16. Narmatha M., Krishnakumar S.V., Study on Android Operating System And Its Versions, *International Journal of Scientific Engineering and Applied Science*, 2, 439–445, **2016**.
17. Android Architecture and Libraries Every Android Developer Should Know. http://simpledeveloper.com/android-architecture/ (Mayıs, **2018**).
18. Arhippainen L., Tähti M., Empirical evaluation of user experience in two adaptive mobile application prototypes, *Proc. of the 2nd International Conference on Mobile and Ubiquitous*

Multimedia, 27-34, **2016**.

19. Heitkötter H., Hanschke S., Majchrzak T.A., Evaluating Cross-Platform Development Approaches for Mobile Applications, *Web Information Systems and Technologies: 8th International Conference*, Porto, Portugal, April 18-21, **2012**.
20. Holzer A., Ondrus J., Trends in Mobile Application Development, Mobile Wireless Middleware, Operating Systems, and Applications - Workshops, Berlin, Germany, 55-64, **2009**.
21. Jabangwe R., Börstler J., Šmite D., Wohlin C., Empirical evidence on the link between object-oriented measures and external quality attributes: A systematic literature review, *Empirical Software Engineering*, 20, 640-693, **2015**.
22. Fenton N.E., Software Metrics: Successes, Failures and New Directions, *Journal of Systems and Software*, 47, 149-157, **1999**.
23. Telegram Applications, https://telegram.org/apps (Nisan, **2018**).
24. Box Plot: Display of Distribution, http://www.physics.csbsju.edu/stats/box2.html (Ağustos, **2018**)
25. Tu H., Sun W., Zhang Y., The research on software metrics and software complexity metrics, *IFCSTA '09 Proceedings of the 2009 International Forum on Computer Science-Technology and Applications*, 1, 131-136, **2009**.
26. Harrison R., Counsell S., Nithi R., Experimental assessment of the effect of inheritance on the maintainability of object-oriented systems, *Journal of Systems and Software*, 52, 173–179, **2000**.
27. Function Point Language Table - QSM SLIM Estimate, http://www.qsm.com/resources/function-point-languages-table (Ağustos, 2018).
28. Feldt R., Magazinius A., Validity Threats in Empirical Software Engineering Research - An Initial Survey, *Proc. of the International Conference on Software Engineering and Knowledge Engineering*, 374–379, **2010**.

ÖZGEÇMİŞ

Kimlik Bilgileri

Adı Soyadı : Sena Sönmez Çiçek

Doğum Yeri : Ankara

Medeni Hali : Evli

E-posta : senasonmezcicek@gmail.com

Adresi : Yenişehir Mah. Silver Home Sitesi C Blok D:16 Pendik - İstanbul

Eğitim

Lise : Sıhhıye Atatürk Anadolu Lisesi

Lisans : İstanbul Teknik Üniversitesi Bilgisayar ve Bilişim Fakültesi Bilgisayar Mühendisliği Bölümü

Yüksek Lisans : Hacettepe Üniversitesi Fen Bilimleri Fakültesi Bilgisayar Mühendisliği Bölümü

Yabancı Dil ve Düzeyi

İngilizce: Dinleme-C1, Okuma-C2, Karşılıklı Konuşma-C1, Sözlü Anlatım-C1, Yazılı Anlatım-C2

Fransızca: Dinleme-B1, Okuma-B2, Karşılıklı Konuşma-B1, Sözlü Anlatım-B1, Yazılı Anlatım-B2

İş Deneyimi

11.2012 – ……

TÜBİTAK Marmara Araştırma Merkezi – Enerji Enstitüsü (MAM-EE): Uzman Araştırmacı

Deneyim Alanları

- Java ile Nesne Yönelimli Programlama (OOP with Java)
- Java'da Spring ve Maven Frameworkleri kullanılarak uygulama geliştirme
- Eclipse ve Spring Tool Suit (STS) kullanarak uygulama geliştirme
- JSF ve PrimeFaces teknolojileri web tabanlı uygulamalar geliştirme
- Tomcat, Apache uygulama sunucularının kullanımı
- SOAP, REST, JSON, XML hakkında bilgi ve tecrübe
- JavaScript, jQuery, HTML, CSS, Ajax vb. web teknolojileri kullanımı
- MSSQL / PostgreSQL veritabanı sorgulama dili kullanımı
- pgAdmin ve Microsoft SQL Server Management Studio Veritabanı Yönetim Ortamı kullanımı
- C/C++ Programlama
- SVN ile Yazılım Versiyon Kontrolü hakkında bilgi ve tecrübe

Tezden Üretilmiş Projeler ve Bütçesi

-

Tezden Üretilmiş Yayınlar

Sönmez Çiçek S, Garousi V, Tarhan A. (2017). Uygulamaların mobil ve masaüstü sürümlerinin kod-tabanlı karşılaştırılması: keşifsel bir çalışma. Ulusal Yazılım Mühendisliği Sempozyumu.

Tezden üretilmiş Tebliğ ve/veya Poster Sunumu ile Katıldığı Toplantılar

-

HACETTEPE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
YÜKSEK LİSANS/DOKTORA TEZ ÇALIŞMASI ORJİNALLİK RAPORU

HACETTEPE ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI BAŞKANLIĞI'NA

Tarih: 03/09/2018

Tez Başlığı / Konusu: **Uygulamaların Mobil ve Masaüstü Sürümlerinin Kod Tabanlı Karşılaştırması: Keşifsel Bir Çalışma**

Yukarıda başlığı/konusu gösterilen tez çalışmamın a) Kapak sayfası, b) Giriş, c) Ana bölümler d) Sonuç kısımlarından oluşan toplam 67 sayfalık kısmına ilişkin, 03/09/2018 tarihinde şahsım/tez danışmanım tarafından *Turnitin* adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı % 3'tür.

Uygulanan filtrelemeler:

1- Kaynakça hariç
2- Alıntılar hariç
3- 5 kelimeden daha az örtüşme içeren metin kısımları hariç

Hacettepe Üniversitesi Fen Bilimleri Enstitüsü Tez Çalışması Orjinallik Raporu Alınması ve Kullanılması Uygulama Esasları'nı inceledim ve bu Uygulama Esasları'nda belirtilen azami benzerlik oranlarına göre tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Gereğini saygılarımla arz ederim.

03.09.2018

Tarih ve İmza

Adı Soyadı:	Sena SÖNMEZ ÇİÇEK
Öğrenci No:	N14127198
Anabilim Dalı:	Bilgisayar Mühendisliği
Programı:	Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı
Statüsü:	☒ Y.Lisans ☐ Doktora ☐ Bütünleşik Dr.

DANIŞMAN ONAYI

UYGUNDUR.

Dr. Öğretim Üyesi AYÇA TARHAN

(Unvan, Ad Soyad, İmza)