İTERATİF YAZILIM GELİŞTİRME İÇİN HATA TAHMİNLEME MODELİ ARAŞTIRMA: BİR DURUM ÇALIŞMASI

INVESTIGATING DEFECT PREDICTION MODELS FOR ITERATIVE SOFTWARE DEVELOPMENT: A CASE STUDY

ANIL AYDIN

YRD. DOÇ. DR. AYÇA TARHAN
Tez Danışmanı

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

2014

Anıl AYDIN'ın hazırladığı "**İteratif Yazılım Geliştirme İçin Hata Tahminleme Modeli Araştırma: Bir Durum Çalışması**" adlı bu çalışma aşağıdaki jüri tarafından **BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Prof. Dr. Hayri SEVER
Başkan .....

Yrd. Doç. Dr. Ayça TARHAN
Danışman .....

Doç. Dr. Pınar KARAGÖZ
Üye .....

Yrd. Doç. Dr. Kıvanç DİNÇER
Üye .....

Dr. Oumout CHOUSEİNOGLOU
Üye .....

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

Prof. Dr. Fatma SEVİN DÜZ
Fen Bilimleri Enstitüsü Müdürü

Sevgili Aileme…

ETİK

Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

20/06/2014

ANIL AYDIN

ÖZET

İTERATİF YAZILIM GELİŞTİRME İÇİN HATA TAHMİNLEME MODELİ ARAŞTIRMA: BİR DURUM ÇALIŞMASI

Anıl AYDIN

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Danışmanı: Yrd. Doç. Dr. Ayça TARHAN

Haziran 2014, 96 sayfa

Yazılım geliştiren kurumların karşılaştığı en büyük problemlerden biri, geliştirme faaliyetleri için gereken kaynakların ve proje sürelerinin belirlenmesi işidir. Yazılım geliştirme esnasında ortaya çıkan hataların sayılarını ve bu hataları düzeltmek için harcanan efor bilgilerini kaydeden kurumlar, ilgili yazılım ürünlerine ait ortaya çıkmamış hataları ve bu hataları düzeltmek için harcanacak eforu doğru bir şekilde tahmin edebilme yeteneğine sahip olurlar. Literatürde, güvenilirlik modelleri aracılığıyla bu tahminleme sürecini gerçekleştiren birçok çalışma yer almaktadır. Ancak, iteratif olarak geliştirilmiş projelere yönelik tahminleme gerçekleştiren çalışma sayısı oldukça azdır. Bu tez kapsamında, iteratif olarak geliştirilen ve hataların ortaya çıktığı geliştirme faz bilgisinin kaydedilmediği bir kamu projesinin, yeni sürümlerine ait hata yoğunluklarını tahminlemeyi hedefleyen bir durum çalışması gerçekleştirilmiştir.

Bu amaçla, ikisi de kendine özgü istatistiksel dağılıma sahip olan Rayleigh Modeli ile Lineer Regresyon Modeli kullanılarak, hem modül hem de proje düzeyindeki hatalılığın iterasyonlar boyunca tahminlenmesi sağlanmıştır. Proje düzeyi için 29 adet; modül düzeyi içinse 15 adet sürüm kullanılarak hata yoğunluklarını tahminleyecek modeller yaratılmıştır. Yaratılan bu modeller ile gerçekleştirilen

tahminleme sonuçları incelendiğinde, hem hata yoğunluklarının gösterdikleri dağılımlar izlenerek hem de tahminlenen değerler ile gerçek değerlerin karşılaştırılması yapılarak elde edilen sonuçlar doğrultusunda, modül seviyesinde Rayleigh Modeli'nin; proje seviyesinde ise Lineer Model'in daha performanslı olduğu ve daha güvenilir sonuçlar ürettiği gözlemlenmiştir.

Bu tez kapsamında, ilgili tahminleme modellerinin yaratılması ve yaratılan bu modeller ile gerçekleştirilen tahminleme işlemlerine ait sonuçların değerlendirilmesi süreci ve bu süreçte takip edilen prosedür adımları, detaylı olarak anlatılmıştır. Çalışmalar esnasında çıkarılan dersler paylaşılarak iteratif geliştirme için tahminleme konusunda çalışma yürütecek kişilere yol gösterici bir kaynağın sunulması sağlanmıştır.

Anahtar Kelimeler: Hata Yoğunluğu Tahminleme, İteratif Yazılım Geliştirme, Rayleigh Modeli, Lineer Regresyon Modeli, Çıkarılan Dersler.

ABSTRACT

INVESTIGATING DEFECT PREDICTION MODELS FOR ITERATIVE SOFTWARE DEVELOPMENT: A CASE STUDY

Anıl AYDIN

Master of Science, Department of Computer Engineering

Supervisor: Asst. Prof. Dr. Ayça TARHAN

June 2014, 96 pages

One of the biggest problems that software organizations encounter is specifying the resources required and the duration of projects. Organizations that record the number of defects and the effort spent on fixing these defects are able to correctly predict the latent defects in the product and the effort required to remove these latent defects. The use of reliability models reported in the literature is typical to achieve this prediction, but the number of studies that report defect prediction models for iterative software development is scarce. In this thesis, we present a case study which was aimed to predict the defectiveness of new releases in an iterative, civil project where defect arrival phase data is not recorded.

With this purpose, we investigated Linear Regression Model and Rayleigh Model, each having their specific statistical distributions, to predict the module level and project level defectiveness of the new releases of an iterative project. The models were created based on defect density data by using 29 successive releases for the project level and 15 successive releases for the module level. Both, the distributions of the defect densities and the comparison results of actual and predicted defect density values shows that at module level the Rayleigh Model and at project level the Linear Regression Model produces more reliable results.

This thesis explains the procedures that were applied to generate the defectiveness models and to estimate the results predicted by these models. By sharing the lessons learned from the studies, it is enabled to provide a guideline for the practitioners who will study about the prediction process for iterative development.

Keywords: Defect Density Prediction, Iterative Software Development, Rayleigh Model, Linear Regression Model, Lessons Learned.

TEŞEKKÜR

Tez çalışmasının gerçekleştirilmesi ve tez metninin hazırlanması esnasında ilgi ve desteğini esirgemeyen, her türlü değerli bilgileri ile hep yanımda olan, çalışmalarımı her zaman destekleyerek ilerlememi sağlayan danışmanım Yrd. Doç. Dr. Ayça TARHAN'a sonsuz teşekkürlerimi sunarım.

Tez aşamasında her türlü değerli bilgisini paylaştığı ve yardımlarını esirgemediği için Nesibe ÖZEN'e teşekkür ederim.

İyi ve kötü günlerimde hep yanımda olan, hiçbir desteğini esirgemeyen bütün dost ve arkadaşlarıma en içten dileklerimle teşekkür ederim.

TÜBİTAK – BİLGEM – Yazılım Teknolojileri Araştırma Enstitüsü'ne(YTE), akademik çalışmalarım için sağladıkları veriler ve yardımlar için teşekkürlerimi bir borç bilirim.

Ayrıca sevgi ve desteklerini hiç esirgemeyen, zor anlarımda hep yanımda olan, bana her zaman güvenen, beni bugünlere getiren canım aileme sonsuz teşekkür ediyorum.

İÇİNDEKİLER

ÇİZELGELER

ŞEKİLLER

SİMGELER VE KISALTMALAR

Simgeler

∑	Toplam sembolü
Π	Çarpım sembolü

Kısaltmalar

IEEE	Institute of Electrical and Electronics Engineers
Std.	Standart
CMMI	Capability Maturity Model Integration
AS	Araştırma Sorusu
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
MMRE	Mean Magnitude of Relative Error
AIAA	American Institute of Aeronautics and Astronautics
KSS	Kod Satır Sayısı
MTTF	Mean Time To Failure
NHPP	Non-Homogeneous Poisson Process
GQM	Goal-Question-Metric
MKA	Metrik Kullanılabilirlik Anketi
MKF	Metrik Kullanılabilirlik Faktörü
KDF	Kümülatif Dağılım Fonksiyonu
PDF	Probability Distribution Function
CDF	Cumulative Distribution Function
SVN	Subversion
COQUALMO	COnstructive QUALity MOdel
ISO/IEC	International Organization for Standardization/International Electrotechnical Commission

1. GİRİŞ

Günümüzde yazılım ürünlerine olan ihtiyacın artmasıyla birlikte, farklı alanlara yönelik geliştirilen yazılım projelerinin sayısında hızlı bir artış meydana gelmiştir. Bu projelerin üçte biri başarılı bir şekilde tamamlanırken; geriye kalan büyük bir kısmı da başarısızlıkla sonuçlanmaktadır [1]. Bu başarısızlığın arkasında yatan nedenlerin başında, kullanılan kaynaklara yönelik etkili bir planlama ve yönetim sürecinin uygulanamaması gelmektedir [2] [3]. Proje yöneticilerinin projelerine ait geleceği görmeleri, proje için gerekli kaynakları uygun şekilde planlamak ve gereksinimlerini tam karşılayan ürünü zamanında teslim etmek açısından önemlidir. Aksi durumda kalite maliyeti artmakta, müşterilerin firmaya olan güveni ve ürünün güvenilirliğine olan inancı azalmaktadır.

Hata maliyeti, ürüne ilişkin kalite maliyetinin en büyük bileşenidir [4]. Yazılım geliştirme sürecinin insan-duyarlı doğası gereği, her yazılım ürünü bir takım hatalar barındırır. Bu nedenle yazılım geliştirme ekibi zamanının büyük bir çoğunluğunu, hatalı kodları düzeltmek için harcar. Bir yazılım ürününün hatalılığının tahmin edilmesi, yöneticinin ilgili projeyi planlama yeteneğini arttıracak ve bunun sonucunda proje takviminde yaşanan gecikmeleri azaltacaktır.

Literatürde, yazılım ürünlerine ait hatalılığı analiz etmeye ve geleceğe yönelik hata tahminleme yapmaya yarayan birçok model önerilmiştir. Makine öğrenme tabanlı modeller bunlardan bazılarıdır. Bu modeller, yazılıma ait problemleri bir öğrenme süreciyle formüle ederek ya da hataları karakteristik özelliklerine göre sınıflandırarak tahminleme yapmayı sağlarlar [5] [6]. Karar ağaçları, yapay sinir ağları, Bayesian ağları ya da kümeleme teknikleri gibi yöntemler en fazla tercih edilen hata tahminleme yöntemlerinden bir kaçıdır [5] [7] [8]. Makine öğrenme haricinde, teknik bilgiler kısmında da detaylı olarak bahsedilecek olan güvenilirlik modelleri bulunmaktadır. Hatalılık tahminleme de en sık kullanılan modeller güvenilirlik modelleridir. Güvenirlilik modelleri, bir yazılım ürününün güvenirliliğini belirlemek ve üründe ortaya çıkabilecek hata sayısını hesaplamak ve yazılım hatalılığını ölçmek amacıyla sık kullanılan modellerdir [9].

Yukarıda bahsedilen hata tahminleme modellerinin pratikte kullanımında bazı zorluklar yaşanmaktadır. İlk olarak, hata kayıtlarındaki bilgi modellerin kullanımına yetmemektedir. Güvenilirlik modellerinin çoğu [9] özellikle test fazında ortaya çıkan

hatalar olmak üzere hataların bulunduğu ve giderildiği faz bilgisini kullanmayı gerektiriyor. Bu bilgi ise ancak kalite yönetiminin önemini kavramış kurumlar tarafından tutuluyor. Pek çok ekip hataları araca girse de faz bilgisini bu kayda girmiyor. Makine öğrenme tabanlı modeller için ise modül ve sınıf seviyelerinde etiketler önemli oluyor [10], ancak bu bilgiler de pek kaydedilmiyor.

Mevcut hata tahminleme modelleri genellikle projeye özgü sonuçlar üretmektedir [11]. Her yazılım projesi; ilgili olduğu alanın, yazılım geliştirme tekniği ya da metodolojisinin, geliştirme yapan ekibin deneyimlerinin farklı olması gibi kendine özgü bir takım özel durumlar içerir. Bunun gibi projeye özel durumlar, ilgili yazılım ürününe ait hata dağılım örüntüsünü etkilemektedir. Bu nedenle, bir yazılım ürününe ait hata tahminleme modeli seçerken ilgili projenin özelliklerine en uygun modelin seçimine dikkat edilmelidir [12]. Bunun için ayrıca, uygun metriklerin seçilmesi ve bu metriklerle başarılı tahminler üretmeyi sağlayacak modellerin kurulması gerekir [13] [14] [15]. Ne var ki mevcut modelleri proje ihtiyaçlarına göre değerlendirmeyi ve model seçimini yönlendiren çok sayıda çalışma bulunmamaktadır. Bunun için ilgili tahminleme modellerinin projeye veya projeyi geliştiren kuruma özel durumlara uyarlanacak şekilde değiştirilmesi ve yeniden derlenmesi gerekmektedir [13] [16].

Hata tahminleme modellerinin de geliştirme sürecine göre uyarlanması gerekmektedir. Bu zorluk aynı zamanda bir önceki problemi (modelin projeye özel sonuçlar üretmesini) destekler niteliktedir. Yazılım geliştirme için Şelale Modeli, Çevik Model ve İteratif Model gibi süreç modelleri kullanılmaktadır. Bu modellerin her biri geliştirme için farklı adım ve pratikler önerir. Örneğin, Şelale Yazılım Geliştirme Modeli, yazılımın; Gereksinim, Tasarım, Geliştirme, Test ve Ortama Kurma gibi fazlardan oluştuğunu ve bu fazların verilen sırada ve sadece bir kez gerçekleştirildiğini söyler. Bu nedenle, bir sonraki faza geçmeden önce üzerinde çalışılan faza ait tüm gereksinim ve ihtiyaçların karşılanmış olması gerektiğini savunur [17] [18] [19]. Bu sav ise bir önceki fazda meydana gelen hatalı durumların giderilmesi için ilgili fazlara dönülmesi esnekliğinin olmadığı anlamına gelir. İteratif Model, Şelale Modeli'nin bu dezavantajını hafifletecek şekilde, birbirini takip eden yazılım sürümlerini kısa sürede geliştirmeyi sağlayan iterasyonlardan meydana gelir [20]. Her bir iterasyonda gelen yeni gereksinimler tanımlanıp bu gereksinimler için geliştirme faaliyetleri en başından ve tekrarlı bir şekilde

uygulanır. Buna göre her bir iterasyon küçük birer Şelale Modeli gibi düşünebilir [21]. Bir iterasyona ait gereksinimleri, sisteme yeni bir özelliğin eklenmesi, güncellenmesi, değiştirilmesi ya da daha önceden geliştirilmiş özelliklerin kaldırılması şeklinde örnekleyebiliriz. İteratif geliştirme yöntemi, Şelale modelinin aksine, daha önceki iterasyonlarda meydana gelen hataları çözebilmek için ilgili faza geri dönebilme esnekliğini sağlar. İteratif Model, bu tekrarların sağladığı avantajlar sebebiyle pek çok kurumunun yazılım ekiplerince uygulandığı halde, bu modeli kullanan projeler için hata tahminleme modeli öneren az sayıda çalışma [20] [21] [22] bulunmaktadır.

Bu tez, yukarıda bahsedilen zorlukları gözeterek; iteratif geliştirme yapan ve hata faz bilgisinin düzenli kaydedilmediği projeler yürüten bir kurumda, hata tahminleme için model araştırmak amacıyla hazırlanmıştır. Dolayısıyla bu tez metni, bahsedilen amaca hizmet eden araştırma sürecini belgelemekte, sonuçları paylaşmakta ve öğretileri tartışmaktadır. Tez çalışması özel olarak, aşağıdaki araştırma sorularını (AS) adreslemektedir:

AS-1: İteratif geliştirilen ve hata faz bilgisi bulunmayan bir projede hangi hata tahminleme model(ler)ini kullanabiliriz?

AS-2: Bu modellerin hata tahminleme performansları nedir ve birbirine kıyasla nasıldır?

AS-3: Hata tahminleme modeli seçimi ve oluşturmayla ilgili öğretilerimiz nelerdir?

Araştırma için yöntem olarak Durum Çalışması (“Case Study”) uygulanmıştır. Çalışma için Runeson ve Host tarafından önerilen durum çalışması kılavuzu [23] temel alınmıştır. Durum çalışmasına konu olan proje, devam etmekte olan bir kamu projesidir. Projeyi geliştiren kurum, ilgili projeye, CMMI-3 olgunluk seviyesinde başlamıştır ve CMMI-4 olgunluk seviyesine geçiş yaparak projeyi geliştirmeye devam etmektedir. Kurumda iteratif olarak geliştirilen projelerin hata dağılım örüntülerinin, iterasyonlar boyunca modül veya proje bazında Lineer Regresyon Modeli ile Rayleigh Modeli’ne uygun dağılımlar gösterdiği gözlemlenmiş ve bu modeller hata tahminleme için seçilmiş ve kullanılmıştır. Bu tezde, Lineer Regresyon Modeli veya Rayleigh Modeli haricindeki diğer güvenilirlik modelleri de incelenmiş olup bu alanda çalışma yapacak diğer kişilere bu modellerden uygun olanının seçilip tahminleme sürecinin nasıl yönetilmesi ve

gerçekleştirilmesi gerektiği anlatılmıştır. Böylece, bir projeye ait hata tahminleme sürecinin nasıl ilerleyebileceği ve hangi adımların izlenebileceğine dair yol gösterici olmak hedeflenmiştir.

Çalışmada, hata yoğunluğunun tahminlenmesi için kullanılacak modellerin eğitilmesi ve kurulması amacıyla proje seviyesinde tüm sürümlere ait hataların %70'i kullanılmıştır. Tüm sürümlerin %30'unu oluşturan diğer sürümlerdeki hatalar ise kurulan modellerinin tahminleme sonuçlarının doğruluğunu kontrol ve test etmek amacıyla kullanılmıştır. Modül seviyesinde ise üç adet modüle ait hatalar örneklem olarak ve kurulacak modellerin eğitilmesi amacıyla kullanılmıştır. İlgili tahminleme modelleri kurulduktan sonra da başka bir modül üzerinde bu model ile tahminleme yapılarak ilgili modülün gerçek hata verileri ile karşılaştırma yapılmış ve modelin doğruluğu kontrol edilmiştir. Kurulan modeller ile gerçekleştirilen tahminleme süreci sonrası elde edilen sonuçların doğruluğunun ve model performansının kontrol edilebilmesi için Karekök Ortalama Hata("Root Mean Square Error" – "RMSE"), Ortalama Mutlak Hata("Mean Absolute Error" – "MAE") ve Ortalama Bağıl Hata("Mean Magnitude of Relative Error" – "MMRE") ölçümleri kullanılmıştır. Burada, proje seviyesinde sürümlere ait hataların %70'inin ve 14 adet modülden 3'ünün eğitim amaçlı seçilmesi başka projelerde de böyle olması gerektiği anlamına gelmemektedir. Bu çalışmada önerilen tahminleme sürecinde bu değerler parametrik olarak tanımlanmıştır ve çalışmayı gerçekleştirecek kişi tarafından artırılıp azaltılabilir.

Çalışmanın ilerleyen kısımları şu şekilde organize edilmiştir: 2. Bölümde, tez kapsamında bahsi geçen ve gerçekleştirilen çalışmada kullanılacak standart, yöntem, teknik ve metotlar hakkında detaylı teknik bilgiler sunulmuştur; 3. bölümde, literatürde yer alan, bu tez kapsamında gerçekleştirilen çalışma ile benzer çalışma gerçekleştiren diğer çalışmalardan bahsedilmiş ve bu çalışmalar ile benzer ve farklı yönler anlatılmıştır; 4. bölümde çalışmanın gerçekleştirilmesi esnasında izlenen araştırma yöntemi ile durum çalışması yöntemleri detaylı olarak anlatılmıştır; 5. bölümde, 4. bölümde bahsi geçen yöntemler doğrultusunda iteratif olarak geliştirilmiş bir projeye ait tahminleme süreci yürütülmüş ve bu tahminleme sonuçları değerlendirilmiştir; 6. bölümde, çalışmanın geçerliliğini etkileyebilecek iç ve dış tehditler anlatılmıştır; 7. Bölümde, bu tez çalışması kapsamında gerçekleştirilen çalışmalar esnasında karşılaşılan zorluk ve engellerden

bahsedilmiş bu zorluk ve engellerin nasıl üstesinden gelindiği anlatılarak tahminleme gerçekleştirecek diğer kişi ve kurumlara yol gösterici bir kaynak sunulmuştur; 8. ve son bölümde ise çalışmaya ait son görüşler paylaşılmış ve ileride gerçekleştirilebilecek çalışmalar hakkında önerilerde bulunulmuştur.

2. TEKNİK BİLGİLER

Yazılım geliştirme süreci boyunca ürüne bir takım hatalar yerleştirilir. Bu hatalar, yazılım ekibi tarafından gözden geçirme ve test gibi kalite güvence faaliyetleri sırasında tespit edilir ve düzeltilir. Ancak, ilgili hataların tespiti ve düzeltilmesi için yazılım ekibinin bir miktar efor sarf etmesi gerekmektedir. Bu kapsamda ilgili hataların yerleştirilmesinden, hataların fark edilmesi ve düzeltilmesi aşamalarına kadar harcanan eforun, zamanın bir fonksiyonu olarak modellenmesi önemlidir. Böyle bir model, geliştirme aşamaları boyunca tespit edilebilecek hataların tahmini ve bu hataları düzeltmede harcanacak zaman ve eforun önceden kestirilip bu duruma göre kaynak planlamasının yapılması için kullanılabilir. Böylece projenin şu anki kalite durumu hakkında çıkarımlarda bulunulabilir ve ilgili önlemlerin alınması sağlanabilir; ileriki sürümler ve hatta projeler hata dağılımı bilgisi kullanılarak planlanabilir [24].

Bu bölümde sırasıyla; yazılım hata tanımı ve hatalılık metriği, yazılım hata tahminleme kavramı, yazılım güvenilirliği ve IEEE Std. 1633, Rayleigh ve Lineer modeller, yazılımda ölçme ile ilişkili model ve standartlar açıklanmıştır.

2.1. Yazılım Hatası ve Hatalılık Metriği

Aşağıda yazılım sistemlerine yönelik hatalılık kavramlarına ilişkin tanımlar verilmiştir [25].

Hata(“Fault”): (1) İnsan kaynaklı hatalar nedeniyle yazılım ürününde ortaya yanlışlıktır. (2) Bir yazılım sisteminde yer alan yanlış bir adım, süreç veya veri tanımıdır. Bir yazılım biriminin kendinden beklenen işlevselliği göstermesine engel olan bir durumdur. Genellikle tasarımda yapılacak değişikliklerle düzeltilebilir. Bir yazılım ürününde yer alan yanlış bir komut buna örnek olarak verilebilir. Hatalar, daha çok yazılımı geliştiren kişiler tarafından fark edilirler.

Bozulma(“Failure”): (1) Bir yazılım sisteminin veya bileşeninin, belirlenen performans gereksinimleri doğrultusunda kendinden beklenen davranışı gösterememesi durumudur. (2) Yanlış sonuç üretilmesi durumudur. Doğru sonuç 10 çıkması gerekirken hesaplanan değerin 12 çıkması buna bir örnek olarak verilebilir. Bozulmalar, daha çok bir sistemin ortama yüklenmesinden önce test edilerek ya da ortama yüklenmesinden sonra müşteri tarafından fark edilmesi sonucu ortaya çıkarlar.

Bozulma Oranı(“Failure Rate”): Verilen bir sürede ortaya çıkan belirli bir kategori ya da önemlilik derecesine sahip bozulmaların sayısıdır.

Bozulma Önem Derecesi(“Failure Severity”): Ortaya çıkan bir bozulmanın yazılıma olan etkisini belirten önem derecesidir.

Yanılma(“Error”): Belirli veya doğruluğu teorik olarak ispatlanmış bir değer ya da durum ile gözlenmiş, hesaplanmış veya ölçülmüş bir değer arasındaki farklılıktır. Gerçek ve hesaplanmış uzunluk değerleri arasında oluşan 30 metrelik bir fark buna örnek olarak verilebilir.

Hata yoğunluğu: Bir yazılım sisteminin birim büyüklüğü başına düşen hata sayısıdır. Hata sayısının, yazılımın büyüklüğüne(örneğin; kod satır sayısı cinsinden büyüklük) bölünmesi ile hesaplanır.

2.2. Yazılım Hata Tahminleme

Hata tahmini; belirli teknikler ve modeller kullanılarak yazılımda tespit edilmiş hataların, bir zaman ekseni üzerinde dağılımlarının oluşturduğu örüntünün gözlemlenip tespit edilebilecek hataların, bu örüntü doğrultusunda tahmin edilmesi işlemidir. İlgili yazılıma ait hataların geçmişteki dağılımı, yazılımın ilerleyen aşamalarında göstereceği hataların dağılımı hakkında bize bilgi verir ve yazılımda kalan hataları tahmin etmemizi sağlar. Tahminlemede kullanılacak modelin belirlenmesi işlemi de bu dağılım gözetilerek yapılır.

Hata tahminleme amacıyla kullanılan teknik ve modeller, kaynak planlamasına da temel olarak, bir yazılım ürünündeki hatalılık oranının ve hataları düzeltmek için gereken efor miktarının tahminlenmesinde kullanılır. Hata dağılımını modellemenin bu kadar önemli olduğu bir durumda, uygun modelin seçilmesi ve kullanılması dikkat edilmesi gereken bir durumdur. Seçilecek modelin bize kullanışlı bilgiler sağlayabilmesi için tahminleme hedeflerine uygun ve kullanılacağı projenin verileri ile uyumlu olması gerekmektedir.

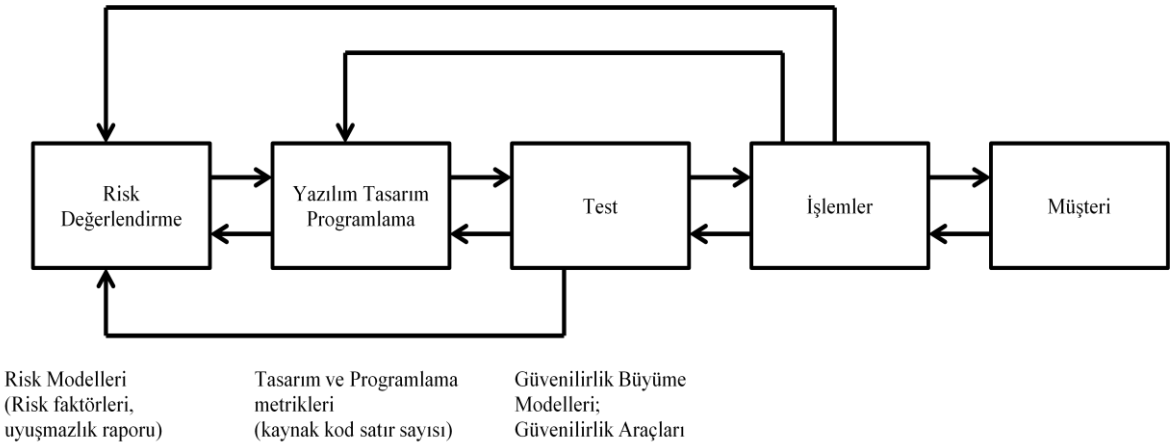
2.3. Yazılım Güvenilirliği ve IEEE Std. 1633

Yazılım güvenilirliği, yazılım kalitesinin en önemli gereklerinden biridir. Yazılım güvenilirliği; belirli bir çalışma ortamında ve belirli bir zaman aralığında, ilgili yazılım sisteminin hatasız olarak çalışabilme yeteneğidir [9]. Bu nedenle yazılım güvenilirliği genellikle, yazılımdaki hata oranıyla ters orantılıdır. Hatalılık

tahminlemede en sık kullanılan modeller güvenilirlik modelleridir. Güvenirlilik modelleri, bir yazılım ürününün güvenirliliğini belirlemek ve üründe ortaya çıkabilecek hata sayısını hesaplamak amacıyla kullanılan modellerdir. Bu hesaplama için dikkat edilecek kriterlerden biri; ay, yıl gibi belirli zaman aralıklarında ortaya çıkacak hata sayısının ve iki hata arasında geçen ortalama zaman verisinin saklanmış olmasıdır [26].

Yazılım güvenilirliğini direkt veya dolaylı olarak ölçmek, değerlendirmek ve sağlamak için, literatürde birçok güvenilirlik mühendisliği modelinden bahsedilmektedir [27] [28] [29]. Ayrıca, yazılım güvenilirliği konusunda geliştirilmiş standartlar yer almaktadır [30] [31] [32]. Bu standartlardan biri olan IEEE 982-2 standardı [33], yazılımın güvenilirliği hakkında bilgi edinebilmek ve ilgili yazılımın güvenilirliğini ölçmek için bir takım metrikler içermektedir. Bir diğer standart ise IEEE 1633 yazılım güvenilirliği standardıdır [9] ve yazılım güvenilirliğinin, yazılım yaşam döngüsü kapsamında değerlendirilmesi ve tahminlenmesi sürecinden bahseder. En çok bilinen güvenilirlik modelleri, kapsamlı bir şekilde, IEEE 1633 standardında anlatılmıştır. Bu bölümün devamında bu standarttan, tez çalışması kapsamında gözden geçirilen veya kullanılan bilgiler özetlenmiştir.

Yazılım güvenilirliği mühendisliği; kurumların, ürün ve süreçlerinin güvenilirliklerinin sağlanması ve iyileştirilmesine yardımcı olan bir disiplindir. AIAA(“American Institute of Aeronautics and Astronautics”), yazılım güvenilirliği mühendisliğini; “Bir sistemin geliştirilmesi esnasında toplanan verilerle sisteme ait güvenilirliğin belirlenmesi, tahminlenmesi ve değerlendirilmesi amacıyla, bir takım istatistiksel tekniklerin uygulanmasıdır.” şeklinde tanımlamıştır. Bu disiplin, bir araya getirilmiş model ve araçlar aracılığı ile yazılım güvenilirliği mühendisliğinin neyi nasıl gerçekleştirdiğini tanımlar. Kurumların yüksek güvenilirlikli projelere sahip olması için bu tarz bir disipline başvurmaları önemlidir. Şekil 2.1 yazılım güvenilirliği mühendisliği süreçlerini özetlemektedir. Şekilde gösterilen oklar yazılım ürününün, bilginin ve/veya hata verilerinin akış yönlerini göstermektedir.



Şekil 2.1. Yazılım Güvenilirlik Mühendisliği Süreçleri

IEEE 1633 standardı, sistem güvenilirliği ya da yazılım güvenliği ile ilgili bilgi vermez; sadece yazılım kalitesi açısından güvenilirliği ele alır. Yazılım güvenilirliği hakkında çalışma yapacaklara tutarlı metot ve modeller sunarak yazılım güvenilirliğini sağlamayı destekler. Bu bilgiyi gereksinimlerin belirlenmesi ve analiz edilmesinden başlayacak şekilde verir. Ayrıca aşağıdakiler için kılavuzluk sağlar:

- Yazılım güvenilirliği risklerinin değerlendirilmesi,
- Daha önceden uygulanmış yazılım süreçlerinin, yazılım güvenilirliğini destekleyecek biçimde mi gerçekleştirildiğinin kararının verilmesi,
- Yazılım tasarım sürecinin iyileştirilmesi, değerlendirilmesi ve yazılım kalitesinin sağlanması,
- Yazılım güvenilirliğinin değerlendirilmesi için kullanılacak prosedürlerin belirlenmesi,
- Test ve model seçimi,
- Yazılım güvenilirliğinin değerlendirilmesi ve tahminlenmesi süreçlerini destekleyecek veri toplama prosedürlerinin seçimi,
- Yeni bir sürümün çıkarılıp çıkarılmayacağı ya da test faaliyetlerinin durdurulup düzeltme faaliyetlerinin gerçekleştirilmesi kararlarının verilmesi,
- İleriye yönelik yazılım güvenilirliğinin tahmin edilmesi (örneğin, bir sonraki hatanın ne zaman oluşabileceğinin hesaplanması),
- Yazılım güvenilirliğinin artırılması için hangi bileşenlerin tekrar tasarlanması gerektiğine karar verilmesi.

2.3.1. Yazılım ve Donanım Güvenilirlikleri

Üretimleri birçok açıdan birbirine benzemekle birlikte yazılım ve donanım ürünleri arasında farklılıklar bulunmaktadır [34] [35]. Bunlar;

- Bir donanım ürünü üzerinde yapılacak değişiklikler bir dizi önemli ve zaman alıcı faaliyetin gerçekleştirilmesini gerektirir. Örneğin; bileşenlerin üretimi, fabrikasyonu, birleştirilmesi, incelenmesi, test edilmesi ve dokümantasyonunun yapılması gibi faaliyetlerin tümü tekrar planlanmalı ve düzeltilmelidir. Ancak, bir yazılım ürününün değiştirilmesi bir donanım ürününe göre daha kolaydır ve genellikle test ve dokümantasyon faaliyetlerinin değiştirilmesine ihtiyaç duyar.
- Yazılım fiziksel olarak var olan bir ürün değildir. İlgili veriyi mantıksal olarak içerir.
- Donanım bir takım fiziksel kısıtlara bağlı kalınarak üretilen, bu nedenle belirli sınırlara sahip bir üründür. Böylece test edilmesi kolay bir ürün haline gelir. Bu, yazılım için geçerli bir durum değildir. Yapılacak çok küçük bir değişiklik bile büyük hatalara neden olabilir. Değişiklikler mantıksal olacağı için belirli sınırları yoktur ve ilgili hatayı test ile bulmak zorlaşır.
- Yazılım hiçbir şekilde zamandan etkilenip yıpranan bir ürün değildir. Yazılımda meydana gelecek hatalar kendini belli etmeyebilir. Diğer yandan donanım ürünlerinde gün geçtikçe performanslarında ve durumlarında bir kötüleşme meydana gelir.
- Yazılım, donanım ürünlerine kıyasla daha karmaşık bir yapıya sahip olsa da donanım ürünlerini çoğaltmak için ihtiyaç duyulan fiziksel şartlara gerek olmadığı için kopyalanmaya müsait bir yapıya sahiptir.
- Bir donanım ürününde yapılacak onarım faaliyetleri, ilgili donanımı eski haline getirebilir. Ancak, bir yazılım ürününde yapılacak değişiklikler, ilgili yazılım ürününün farklı bir duruma geçmesine (farklı bir ürüne dönüşmesine) neden olur.
- Donanım güvenilirliği, daha çok saat zamanı ile ifade edilirken; yazılım güvenilirliği, işletim süresi ve takvim zamanı ile (kronolojik) ifade edilmektedir.

2.3.2. Yazılım Güvenilirliği ile İlgili Bazı Tanımlar

Bozulmalar Arası Ortalama Zaman("Mean Time To Failure" – "MTTF"): Bir sistemin herhangi bir bozulma meydana gelmeden çalışabildiği ortalama süredir. Bir sistemin güvenilirliğinin değerlendirilmesi için kullanılan yöntemlerden biridir.

Hata Toleransı: Bir yazılım ürününün kendinden beklenen işleri, bir hata ortaya çıkana kadar gerçekleştirebilmesi durumudur.

Güvenilirlik Riski: Gereksinimlerdeki değişikliklerin güvenilirliği düşürmesi olasılığıdır.

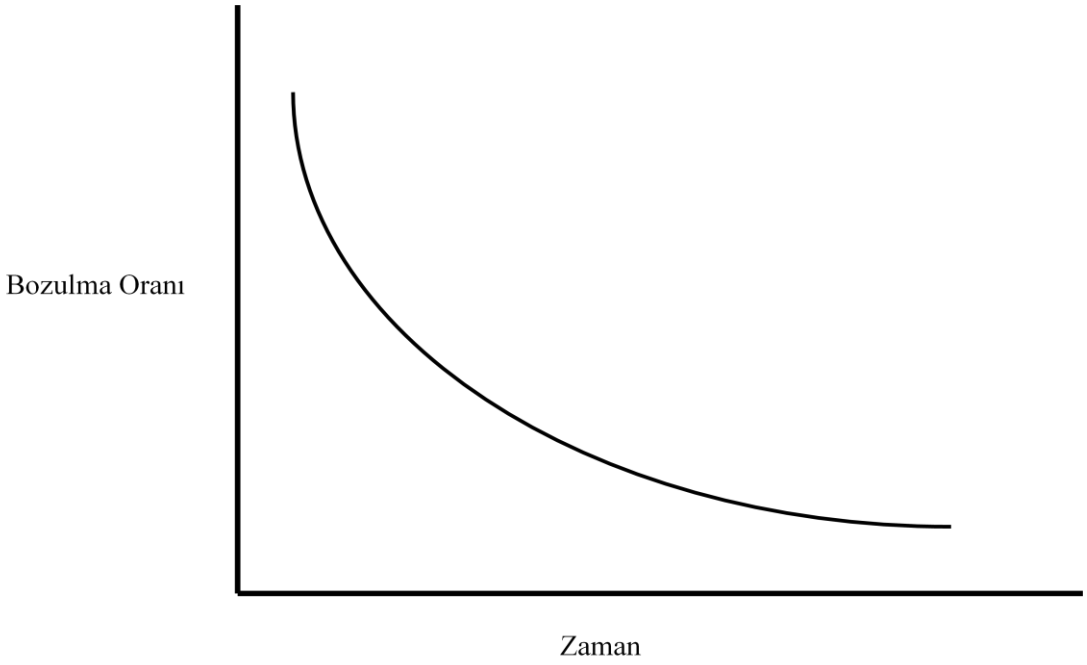
Yazılım Kalitesi: Bir yazılım ürününün tüm özellikleri ile müşterinin beklentilerini karşılayabilme derecesidir.

2.3.3. Yazılım Güvenilirliğinin Modellenmesi

Yazılım, doğası gereği soyut olması nedeniyle hatalılığa çok yatkın bir üründür. Bu hatalılığı azaltmak amacıyla geliştirme aşamasında test, gözden geçirme, izleme gibi faaliyetler ile bir takım metrikler toplanarak hataların tespiti ve düzeltilmesi sağlanır. İlgili hataların yönetilebilmesi için hataların tanımlanması, önem ve kritiklik seviyesine göre sınıflandırılması ve yazılımda kalan ve fark edilmemiş hataların etkilerinin gözlemlenmesi amacıyla bazı modellerin uygulanması gerekir.

Yazılımda bulunan hataların düzeltilmesi ile yazılımın hatalılık oranında azalma meydana gelmektedir. Şekil 2.2'de bu düşüşü gösteren bir grafik yer almaktadır. Bu grafik incelendiğinde herhangi bir zamanda ilgili yazılım ürününe ait hatalılık oranı değerleri elde edilebilir ve seçilecek bir istatistiksel model ile ilgili hatalılık oranı verileri kullanılarak bir tahminleme mekanizması kurmak mümkün hale gelir. Bu ölçümün yapılmasının iki ana nedeni bulunmaktadır:

- Hedeflenen güvenilirliği sağlamak için ihtiyaç duyulacak ekstra zamanın belirlenmesi,
- Ürüne ait sürüm alınmadan önce beklenen güvenilirliğin tanımlanması



Şekil 2.2. Yazılım Güvenilirliği Bozulma Oranı Eğrisi

Bu grafik geliştirme platformlarının, yazılım süreçlerinin ve tasarım ortamlarının belirli bir olgunluk seviyesinde kalması ya da ilgili ortamların yazılım kalitesini artıracak biçimde iyileşmesi doğrultusunda geçerli olacaktır. Eğer, geliştirme ortamlarının kötüleşmesi, geliştirme ekibinin belirli uzmanlık düzeyinin altına gerilemesi gibi durumlar ortaya çıkarsa, yazılımda meydana gelecek hataların sayısı artar, hatalar düzeltilemez duruma gelir ya da hatalar düzeltilse bile başka hataların oluşmasına neden olunur. Bu nedenle ilgili yazılım ürününün maliyeti artacağından, bakım faaliyetleri yürütülemez ve ilgili yazılım ürünü kullanılamaz hale gelebilir. Bu gibi durumlarda yukarıdaki grafik anlamını kaybeder.

Güvenilirliğin değerlendirilmesi ve tahminlenmesi, bir takım kısıtlar çerçevesinde mümkündür. Güvenilirliğin belirli bir doğruluğa sahip olabilmesi için kullanılacak verilerin değerlendirme ve tahminleme faaliyetlerini destekleyici bir niteliğe sahip olması gerekir. İyi veri demek doğru veri yani verilerin doğru olarak kaydedilmesi demektir. İyi veri ayrıca, uygun veri yani ilgili ihtiyaçları karşılayacak doğrultuda toplanmış veri anlamına gelmektedir. Bu kısıtlar dâhilinde güvenilirlik analizi ve tahminleme doğru bir biçimde gerçekleştirildiği takdirde fayda sağlayacaktır. Yine de aşağıda bahsedilen birkaç faktör nedeniyle tahminleme modellerinden faydalanım olumsuz etkilenebilmektedir:

- Bozulma kriterindeki (örneğin; hangi durumların bozulma olarak kabul edileceği) değişiklikler,
- Test edilme aşamasındaki bir üründe yapılacak büyük değişiklikler,
- Yazılımın çalıştığı ortamda yapılacak önemli değişiklikler

Yukarıda bahsedilen faktörler nedeniyle değerlendirme ve tahminleme modellerinin parametrelerinde bir takım değişiklikler meydana gelir ve bu da ilgili modelde değişiklik yapılmasını zorunlu kılar. Gereken değişiklikler yapılsa bile kullanılan model eski performansını kaybetmiş olacaktır. Bu nedenle, bir güvenilirlik modeli seçimi yapılırken ilgili kurumsal politikaların da dikkate alınması gerekmektedir.

2.3.4. Yazılım Güvenilirliği Değerlendirme ve Tahminleme Prosedürleri

IEEE 1633 standardı, yazılım güvenilirliğinin değerlendirilmesi ve sonuçların analiz edilmesini sağlayan ve 13 adımdan oluşan bir prosedür önermektedir. Uygulanacak adımlar, projeye ve mevcut yaşam döngüsüne göre uyarlanabilir bir yapıya sahiptir.

a) **Uygulamanın tanımlanması:** Yazılım güvenilirliği değerlendirme ve tahminleme sürecine dâhil edilen tüm yazılım bileşenlerinin tanımının yapılması gerekir.

b) **Güvenilirlik gereksinimlerinin belirlenmesi:** Tüm yazılım bileşenlerinin birbirleriyle olan ilişkileri belirlenmelidir. Çünkü yazılım sistemi bir bütündür ve eğer bir bileşen başarısızlığa uğrarsa tüm sistem başarısız olur.

c) **Güvenilirlik gereksinimlerinin ayrıştırılması:** Güvenilirlik gereksinimlerinin ilgili yazılım bileşenlerine paylaştırılması gerekmektedir. Tüm yazılımın güvenilirliği ilgili yazılım bileşenlerinin güvenilirliklerinin çarpımına eşittir.

$$R_t = \prod_{i=1}^{n} R_i \tag{2.1}$$

Ayrıca bir bileşenin kritikliği ile bu bileşenin hatasız çalışma süresi arasındaki ilişkiyi düşünecek olursak, bir bileşenin kritikliği arttıkça hatasız çalışma süresi de kısalacaktır.

d) **Güvenilirlik risk değerlendirmesinin yapılması:** Güvenilirlik risklerinin değerlendirilmesi sürecinde dikkat edilmesi gereken en önemli faktör gereksinim ya da gereksinim değişikliklerinden kaynaklanan yazılım hatalarıdır. Risk değerlendirmesi, kalan bozulmaların neden olacağı riskler ile bozulmalar arası geçen sürede ortaya çıkan riskler hesaplanarak yapılır.

e) **Hata tanımlarının yapılması:** Daha önceden yapılan yanılma("error"), hata("fault") ve bozulma("failure") tanımlarına göre projeye özel hata tanımları yapılır. Test sürecine başlamak için bu tanımlar üzerinde; geliştiriciler, testçiler ve kullanıcılar arasında bir anlaşmaya varılmış olmalıdır. Önemli olan proje süresi boyunca bu tanımların tutarlı olmasıdır. Projede ortaya çıkan hatalar önem derecesine göre sınıflandırılmalıdır.

f) **İşletim ortamının tanımlanması:** Projenin çalışacağı ortamın; sistem yapılandırma ayarları, sistemin evrimi ve sistemin çalışma profiline göre nitelendirilmesi gerekir. Sistem yapılandırma ayarları açısından her bir bileşenin ve bu bileşenlerin bir araya gelmesiyle meydana gelen sistemin güvenilirliğinin dikkate alınması gerekmektedir. İlgili sistem, test edilip yeni kodlar ya da bileşenler eklendikçe değişime uğrar. Sistem, bazı çalışma modları içerir. Bunlar, kullanıcıya veya uygulamanın yüklendiği ortama göre değişiklik gösterebilir. Tüm bu modların güvenilirliği ayrı ayrı ele alınmalıdır.

g) **Testlerin seçilmesi:** Test ve test durumları, sistemin esas çalışma şekline uyacak biçimde seçilmelidir.

h) **Modellerin seçilmesi:** Bir ya da birden fazla güvenilirlik modeli seçilmeli ve uygulanmalıdır. Literatürde birçok güvenilirlik modeli yer almaktadır, ancak bu modellerin tüm ortamlara uyumlu olacağının bir garantisi yoktur. Bunun için kullanıcı ortamı üzerinde modeller denenerek karşılaştırmalar yapılmalı ve en uygun modeller seçilmelidir.

i) **Verilerin toplanması:** Toplanan veriler güvenilirlik hedeflerini karşılayacak nitelikte ve yeterlilikte olmalıdır. Bunun için; veri toplama hedefleri açıkça belirlenmeli, uygun veri toplanmalı ve toplanan verinin gözden geçirilerek ihtiyaçlara cevap verir nitelikte olduğuna kanaat getirilmelidir. Bu süreçte bazı adımların uygulanması gerekir. Bu adımlara örnek verecek olursak; hedefler belirlenmeli, veri toplama süreci için bir plan yapılmalı, uygun veri toplama araçlarından faydalanılmalı, gerekli eğitimler alınmalı, veri toplama süreci planı uygulanmalı, ilgili veri toplama süreci izlenmeli, toplanan veriler

sürecin devamlılığını sağlıyor mu kontrolü yapılmalı ve ilgili tüm birimlere geri bildirim verilmelidir.

j) **Parametrelerin hesaplanması:** İlgili parametrelerin hesaplanması için uygun bir metot seçilmelidir. Bunun için birçok metot bulunmaktadır. Maksimum benzerlik ("maximum likelihood") hesaplama en çok tercih edilen yöntemdir.

k) **Modelin geçerlenmesi:** İlgili süreçte seçilen model ya da modellerin geçerlenmesi işleminin gerçekleştirilmesi gerekmektedir. Model geçerleme süreci, ayrıca, veri toplama ve parametre hesaplama işlemlerini de kapsamalıdır. Modelin geçerlenmesi için gerekli veriler; projenin hata geçmişinden, projenin prototipinden ve benzer projelere ait bilgilerden toplanabilir. Bu verileri toplayan analistler ilgili modelleri belirli sıklıklarla çalıştırarak model sonuçlarını gerçek verilerle karşılaştırmalıdır. Karşılaştırma için belirli bir tahminleme kriteri ve uyum derecesini ("goodness-of-fit") test eden geleneksel bir istatistiksel yöntem (Örneğin chi-square ya da Kolmogorov-Smirnov (K-S)) kullanılabilir. Bir model seçildikten sonra ilgili modelin devamlı olarak geçerlenmesi önemli bir konudur.

l) **Değerlendirme ve tahminlemenin analizi:** İlgili veri toplandığında ve model parametreleri hesaplandığında analize başlanabilir. Bu analiz, yazılımın var olan güvenilirliğini değerlendirme, kodda kalan hataların sayısını ya da testlerin tamamlanma süresini tahmin etme işlemlerini içermelidir. Bu adım, iki alt adımı içermektedir:

- **Var olan güvenilirliğin değerlendirilmesi:** Test ve işletimde gerçekleştirilen güvenilirlik değerlendirme, temelde aynı prosedürü takip eder. Yine de küçük bir fark bulunmaktadır. Test süreci boyunca fark edilen hatalar olabildiğince erken bir biçimde düzeltilebilir. Böylece sistemin güvenilirliğinde gözle görülür bir artış yaşanır. Ancak, işletimsel fazda fark edilen hataların bir an önce düzeltilme imkânı yoktur. Değişikliklerin yansıtılabilmesi için yeni bir sürüm alınana kadar beklenmelidir.
- **Güvenilirlik hedefinin başarımının tahminlenmesi:** Güvenilirlik hedefinin tahminlenmesi için bir model kurulmalıdır. Kurulan

tahminleme modeli gerçek verilere uyumu açısından incelenmeli ve modelde gerekli iyileştirmeler yapılmalıdır.

m) <u>Test için gerekli ek sürenin öngörülmesi:</u> Başlangıç ve hedef hata yoğunlukları ve model parametreleri bilindiği takdirde test süreci için gereken ek süre tahmin edilebilir.

2.3.5. Yazılım Güvenilirliği Tahminleme Modelleri

Yazılım güvenilirliği tahminleme için modeller, üç genel sınıfa ayrılmaktadır. Bunlar; Üssel NHPP(Non-Homogeneous Poisson Process) Modelleri, Üssel Olmayan NHPP Modelleri ve Bayesian Modelleri'dir.

Üssel NHPP Modelleri: Üssel NHPP modelleri, olasılıksal süreçlerden ve risk fonksiyonu yaklaşımından faydalanır. Risk fonksiyonu, z(t), genellikle işletimsel zaman olan t süresinin bir fonksiyonudur. Başarım olasılığını gösteren bir fonksiyon olan güvenilirlik fonksiyonu R(t),

$$R(t) = e^{\left[-\int_0^t z(y)\,dy\right]} \tag{2.2}$$

eşitliği ile verilmektedir. Bazen de güvenilirlik fonksiyonu eşitliği MTTF cinsinden şöyle ifade edilebilir,

$$MTTF = \int_0^\infty R(t)\,dt \tag{2.3}$$

MTTF'nin tanımlı olmadığı durumlarda risk fonksiyonu tercih edilir. Bu model sınıfında;

- Schneidewind modeli,
- Shooman modeli,
- Temel Musa modeli,
- Jelinski ve Moranda modeli,
- Genelleştirilmiş üssel model

yer almaktadır.

Üssel Olmayan NHPP Modelleri: Üssel olmayan NHPP modelleri de olasılıksal süreçler ve risk fonksiyonu yaklaşımını kullanır. Genellikle test süreci tamamlandığı zaman uygulanır. Bu model sınıfında;

- Duane modeli,
- Brooks ve Motley'in Binomial ve Poisson modelleri,
- Yamada'nın S-shaped modeli,
- Musa ve Okumoto'nun logaritmik Poisson modeli

yer alır.

Bayesian Modelleri: NHPP modelleri, risk fonksiyonunun programdaki hata sayısı ile doğru orantılı olduğunu varsayar. Bu nedenle güvenilirlik, hata sayısının bir fonksiyonudur. Bayesian yaklaşımı, ilgili programın kullanılmayan kısımlarında birçok hatanın bulunabileceğini ve bu nedenle programın sık kullanılan ve daha az hata içeren kısımları ile yüksek güvenilirlikte çalışıyormuş gibi görünebileceği durumunu ele alır. Bu model sınıfı; Littlewood tarafından geliştirilen modelleri içerir.

2.3.6. Yazılım Güvenilirliği Verileri

IEEE 1633 standardının güvenilirlik verilerinin toplanması için tanımladığı 9 adet adımdan oluşan bir prosedür bulunmaktadır.

Adım 1: Veri toplama planı hedeflerinin belirlenmesi

Bu adım; veri toplama hedefleri, veri gereksinimleri ve toplanacak veri öğelerini içeren veri toplama planının oluşturulması adımıdır. Veri toplama süreci maliyetli bir süreç olması nedeniyle, ilgili plan hazırlanırken toplanacak verinin mal olacağı maliyete değip değmeyeceğinin kararının iyi verilmesi gerekmektedir.

Adım 2: Veri toplama süreci planının hazırlanması

Veri toplama sürecinin planlanması aşamasında, sistemde rol alacak tasarımcı, yazılımcı, testçi, kullanıcı ve yönetim ekibi gibi tüm birimlerin dâhil olması gerekir. Veri toplama sürecinin hedefleri ortaya konur ve başlangıç aşamasında kullanılmak üzere bir plan hazırlanır. Bu planda; hangi veri öğelerinin toplanacağı, bu verileri kimin toplayacağı, verilerin ne sıklıkta raporlanacağı, raporlanacak verilerin formatları, verilerin nasıl saklanacağı ve işleneceği ile veri bütünlüğünü sağlamak için verinin nasıl izleneceği gibi konular belirlenir. Verilerin daha sonraki

kullanım amaçları düşünülerek nasıl kaydedileceğine dair prosedürler dikkatli bir biçimde hazırlanır ve veriyi toplayacak proje üyelerinin harcayacağı efor miktarını azaltacak ve veri güvenilirliğini sağlayacak teknik, yöntem ve araçlar incelenir.

Adım 3: Veri toplama araçlarının kullanılması

Veri toplama esnasında harcanacak efor miktarını azaltmak ve toplanan verinin güvenilirliğinin ve tutarlılığının sağlanabilmesi için mümkün olduğunca veri toplama araçlarından faydalanılması ve sürecin otomatikleştirilmesi uygun olacaktır. İlgili veri toplama sürecinin otomatikleştirilebilir olup olmadığını kontrolünde bir takım sorulara cevaplar aranmalıdır. Bunlar; ilgili aracın elde edilebilirliği yani aracın satın alınmasına ya da geliştirilmesine karar verilmesi, ilgili aracın satın alınma ya da geliştirilmesi esnasında projeye mal olacak maliyete karar verilmesi, ilgili aracın ne zaman elde edilebileceği yani veri toplama süreci ile takvimsel olarak uyuşup uyuşmadığının kararının verilmesidir.

Adım 4: Eğitim sağlama

İlgili veri toplama aracına karar verildikten ve sürecin tanımı yapıldıktan sonra veri toplama süreci ile ilgili tüm birimlerin ilgili aracın kullanımı açısından eğitilmesi gerekmektedir. Veri toplayacak kişilerin verinin hangi amaçla ve hangi verileri toplayacağını iyi anlaması gerekir. Böylece süreç performansının artırılması sağlanabilir.

Adım 5: Sürecin denenmesi

Belirlenen veri toplama sürecinin herhangi bir problem ya da yanlış anlaşılma içermediğinin anlaşılması için denenecek şekilde işletilmesi gerekmektedir. Böylece sürece başlanmadan önce varsa süreçteki problemler belirlenip en kısa sürede giderilecektir.

Adım 6: Planın yürütülmesi

İlgili veriler geliştirilen plan doğrultusunda hızlı bir biçimde toplanıp gözden geçirilmelidir. Ayrıca, ileride analiz yapılabilmesi için ortaya çıkan beklenmedik sonuçların raporlanması gerekmektedir.

Adım 7: Veri toplama sürecinin izlenmesi

Veri toplama sürecinin, belirlenen hedefleri ve güvenilirlik hedeflerini karşılayıp karşılamadığının kontrolünün yapılması gerekmektedir. Bunun için de veri toplama aktivitelerinin izlenmesi gerekir.

Adım 8: Toplanan veriler ile tahminlemenin yapılması

Yazılımın güvenilirliğinin takip edilebilirliğinin maksimum seviyeye çıkarılabilmesi amacıyla toplanan veriler ile belirli sıklıktaki aralıklarla tahminleme yapılması gerekmektedir.

Adım 9: Geribildirimin sağlanması

Veri toplama sürecinin uygulanmasında karşılaşılan başarılı ve başarısız durumlar göz önüne alınarak gerekli düzenlemelerin bir an önce yapılabilmesi için gerekli geri bildirim mekanizmasının oluşturulması gerekmektedir.

2.4. Rayleigh Modeli

Rayleigh Modeli, tüm geliştirme aşamasını modelleyen parametrik bir modeldir. İstatistiksel dağılımlara dayanır ve Weibull dağılımı ailesinin bir üyesidir. Bu ailedeki modeller birçok mühendislik alanında güvenirlik analizi yapmak amacıyla kullanılır. Rayleigh Modeli aslında, Weibull dağılımının özelleşmiş bir halidir [36].

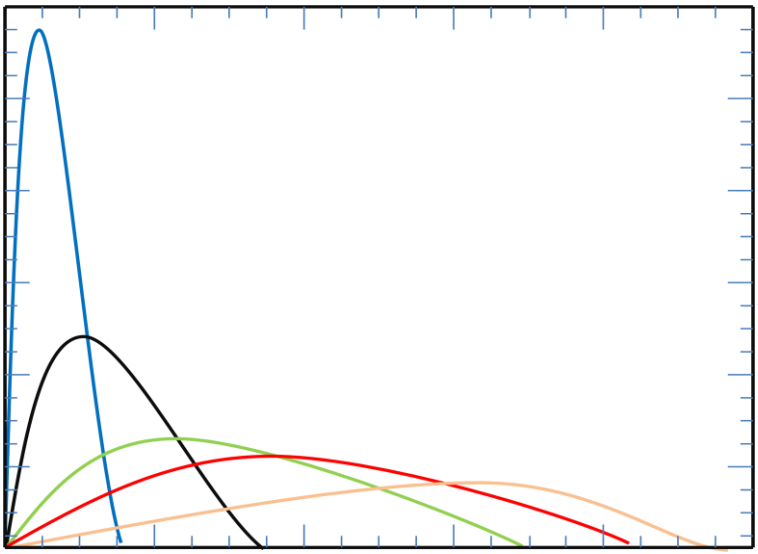
Eğer bir yazılım sistemine ait hataların dağılımı, Rayleigh dağılımı gösteriyorsa ilgili sistemin ortaya çıkmamış hataları, Rayleigh Modeli kullanılarak tahminlenebilir. Böylece Rayleigh Modeli'ni, herhangi bir zaman periyodunda ortaya çıkabilecek hata sayısını, hata yoğunluğunu ya da hata düzeltme faaliyetleri için harcanan eforu tahmin etmede kullanabiliriz. Tahminlediğimiz sonuçlarla gerçek değerleri karşılaştırarak ilgili projenin gidişatı hakkında yorum yapabiliriz.

Rayleigh Modeli, hataların dağılımını tahminlemek için iki adet fonksiyon kullanır. Bunlar; Olası Dağılım Fonksiyonu ("Probability Distribution Function - PDF") ve Kümülatif Dağılım Fonksiyonu ("Cumulative Distribution Function – CDF")' dur.

Olası Dağılım Fonksiyonu: Bu fonksiyon, bir yazılım ürününde belirli bir zaman diliminde ortaya çıkan hataların sayısını tahmin etmek için kullanılır. Fonksiyona ait eşitlik şöyledir:

$$f(t) = K\ x\ (2t/c^2)\ x\ e^{-(t/c)^2} \quad (2.4)$$

Şekil 2.3'de görüldüğü üzere Olası Yoğunluk Fonksiyonu'nda eğrinin kuyruğu zaman arttıkça sıfıra yaklaşmaktadır fakat sıfır olamaz. Eğrinin altında kalan alanın toplamı 1'e eşit olmaktadır. Ayrıca, görüldüğü gibi zaman ekseninde (x) ilerledikçe hata sayısında bir azalma meydana gelmektedir. Hatalar fonksiyona göre zamana yayılmaktadır.

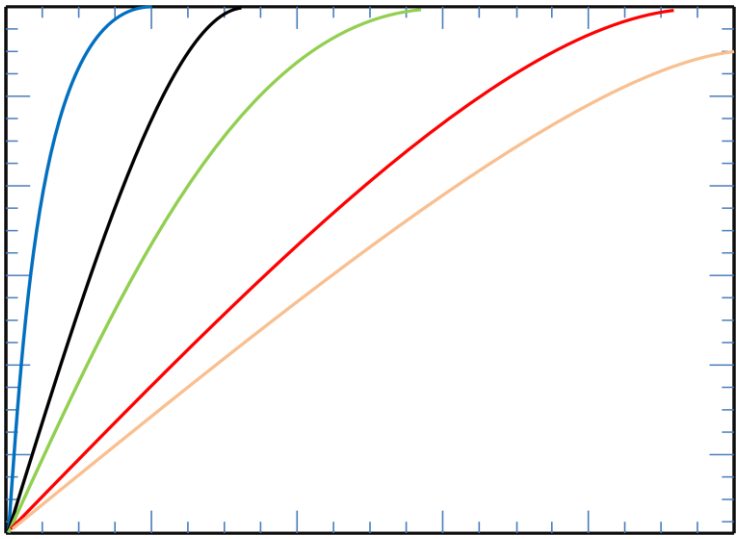


Şekil 2.3. Rayleigh Modeli Olası Yoğunluk Fonksiyonu Dağılımları

Kümülatif Dağılım Fonksiyonu: Bu fonksiyon, bir yazılım ürününde belirli bir zaman diliminde ortaya çıkan toplam hata sayısını tahmin etmek için kullanılır. Fonksiyona ait eşitlik şöyledir:

$$F(t) = K\ x\ (1 - e^{-(t/c)^2}) \quad (2.5)$$

Şekil 2.4'deki grafikte hataların zamana göre toplanarak nasıl bir dağılım gösterdiği görülebilir.



Şekil 2.4. Rayleigh Modeli Kümülatif Dağılım Fonksiyonu Dağılımları

Rayleigh Modeli'ne ait bu iki fonksiyonda yer alan "c" parametresi, Rayleigh eğrisinin tepe ("peak") noktasına eriştiği zamanı gösteren tmax parametresi ile ilişkilidir. Bu iki parametre arasındaki ilişki şu şekilde gösterilir:

$$c = t_{max}\, x\, \sqrt{2} \tag{2.6}$$

Yine her iki fonksiyonda da yer alan "K" parametresi, ilgili yazılım ürününe ait toplam hata sayısını gösteren bir parametredir. F(t) kümülatif dağılım fonksiyonu, t anındaki kümülatif hata sayısını gösteren fonksiyondur. F(t) fonksiyonunun eşitliği kullanılarak tmax anındaki hata sayısına bakacak olursak, bu hata sayısı tüm hataların toplamının %40'ına eşittir [37]. Bu öneriyi destekleyen eşitlik şöyledir:

$$100\, x\, \left(\frac{F(t_{max})}{K}\right) = 100\, x\, \left(1 - e^{-(0,5)}\right) \cong \%40 \tag{2.7}$$

%40 kuralını temel alarak gerekli hesaplamaları yapabiliriz. Örnek olarak; tmax=7 zamanında ortaya çıkan toplam hata sayısının 429 olduğunu farz edelim. Buna göre toplam hata sayısı;

$$429\, x\, \left(\frac{100}{40}\right) \cong 1072 \tag{2.8}$$

K ve tmax değerleri bilindiği için, ilgili eşitlikler kurularak ve fonksiyonlardaki t parametresi yerine grafiğin x ekseninde yer alan zaman bilgisi yerleştirilerek herhangi bir t anındaki hata ve kümülatif hata sayısı değerleri için gerekli

hesaplamalar yapılabilir, olası dağılım fonksiyonu ve kümülatif dağılım fonksiyonu grafikleri çizilebilir.

İlgili değerleri kullanarak gerekli hesaplamaları yapabiliyoruz. Ancak, var olan verilerimizin bu hesaplamaya ne kadar uygun olduğu konusu önemlidir. Rayleigh Modeli, görüldüğü üzere hata sayısı tahmini yapmaya olanak verdiği gibi proje boyunca tespit edilen hataların yoğunluğu (hata sayısı/toplam kod satır sayısı) hakkında ve hataları düzeltmek için harcanan efor hakkında tahmin yürütmeye de olanak vermektedir. Bunun için hata yoğunluğu ve hata düzeltme eforu verilerine ait dağılımların, Rayleigh dağılımına uygunluk göstermesi gerekir. Rayleigh Modeli ile hata düzeltme efor tahmini yapabilmek için ayrıca, her bir hataya ait hata düzeltme eforu bilgisinin tutulmasına ihtiyaç vardır. Ek olarak, dağılımın izleneceği grafikte yatay eksende yer alan zamanın belirli ve düzgün aralıklara bölünmüş olması gerekmektedir. Bu çalışmada projeye ait hata yoğunluğu verileri üzerinden tahminleme gerçekleştirilecektir.

Yazılım kalitesini modellemek için kullanılan Rayleigh dağılımı iki ana varsayıma dayanır. Bunlar;

- Rayleigh dağılım eğrisi ne kadar yüksekse hatalılık yani hata sayısı da o kadar yüksektir.
- Geliştirme aşamasında erken hata tespiti yapılır ve hatalar düzeltilirse sonraki aşamalarda daha az hata ile karşılaşılır.

2.5. Lineer Model

Regresyon analiz modeli, bağımlı ve bağımsız değişkenler arasındaki ilişkiyi tanımlayan istatistiksel bir modeldir [38].

Bu modelin amacı;

- Bağımsız değişkenlerdeki değişikliklere göre bağımlı değişkene ait değerleri hesaplamak ve tahmin etmek,
- Bağımsız değişkenlerle bağımlı değişkenler arasındaki ilişkiyi analiz etmektir.

Bağımlı değişkene ait değerlerin hesaplanması için bir veya birden çok bağımsız değişkene ait değerler kullanılır. Eğer modele ait sadece bir bağımsız değişken bulunuyorsa ilgili model, Basit Regresyon Modeli olarak adlandırılır. Basit

Regresyon Modeli, ‘x’ ve ‘y’ boyutları arasındaki ilişkiyi belirli bir eğime sahip düz bir doğru çizerek gösterir; verilen bir takım veri noktalarına minimum ortalama uzaklıkta çizilen bu doğruya ait fonksiyonun bulunmasını hedefler. Böylece tüm veri kümesine en yakın değerler hesaplanarak tahminleme hata oranını en aza indirecek Lineer Regresyon fonksiyonu bulunmuş olur. İlgili noktalar kümesinin doğruya olan uzaklıklarının karelerinin toplamı ne kadar küçükse oluşturulan Lineer Regresyon Modeli de bir o kadar uygundur ve doğru sonuçlar üretir. Bağımlı ve bağımsız değişkenler arası ilişkiye örnek olarak, yapılan aylık ev masraflarının evde yaşayan kişi sayısı ile olan ilişkisi verilebilir. Burada evde yaşayan kişi sayısı bağımsız; yapılan aylık ev masrafları ise bağımlı değişkendir. Çünkü evde yaşayan kişi sayısına göre aylık ev masrafı değeri değişkenlik gösterecektir. Yani aylık ev masrafları evde yaşayan kişi sayısına bağlıdır.

Modelin varsayımları şunlardır;

- Bağımsız değişken ‘x’ ile bağımlı değişken ‘y’ arasındaki ilişki lineerdir.
- ‘x’ değerleri hatasız bir biçimde ölçülmüş, rastgele veya tahminle elde edilmemiş değerlerdir.

Basit Lineer Regresyon Modeli’ne ait fonksiyon eşitliği şöyledir;

$$y = a + bx \tag{2.9}$$

Burada ‘x’ açıklayıcı ya da bağımsız değişken; ‘y’ ise bağımlı değişken olarak adlandırılır. ‘b’ çizilen grafiğe ait çizginin eğimine eşittir. ‘a’ değeri, x=0 olduğunda grafiğin y eksenini kesen noktasındaki değerdir.

Verilen bir takım ‘x’ ve ‘y’ değerler kümesine göre ‘a’ ve ‘b’ değerlerinin hesaplanışı şöyledir [39];

$$b = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{2.10}$$

$$a = \bar{y} - b\bar{x} \tag{2.11}$$

Yukarıda verilen formülde ‘$\bar{x}$’ değeri, kullanılan veri kümesindeki ‘x’ değerlerinin ortalaması; ‘$\bar{y}$’ değeri ise kullanılan veri kümesindeki ‘y’ değerlerinin ortalamasıdır.

Bu modelin en büyük dezavantajı, lineer bir dağılım göstermeyen ya da lineere yakın olmayan dağılımlara ait değerlerin hesaplanmasında ve tahminlenmesinde

ortaya çıkan sonuçların yanılma payının yüksek olmasıdır. Lineer dağılıma uzak bir noktalar kümesi için, oluşturulan Lineer Model'in yaptığı hesaplama sonucu ortaya çıkan değerler gerçek veri kümesine uzak olacaktır.

2.6. Yazılımda Ölçme

Her mühendislik dalında olduğu gibi yazılım geliştirme alanında da mühendislik faaliyetleri ile üretilen verileri elde ederek değerlendirebilmek ve bu verilerden mühendislik yönetimini iyileştirmeye yarayacak bilgileri elde edebilmek için ölçmeye ihtiyaç duyarız. Ölçme, var olan sistemlerimizin gelişmesinde büyük rol oynamaktadır. Ancak, bu işlemi doğru bir biçimde gerçekleştirebilmek için bir takım standartlar ve kurallar kullanmak zorundayız.

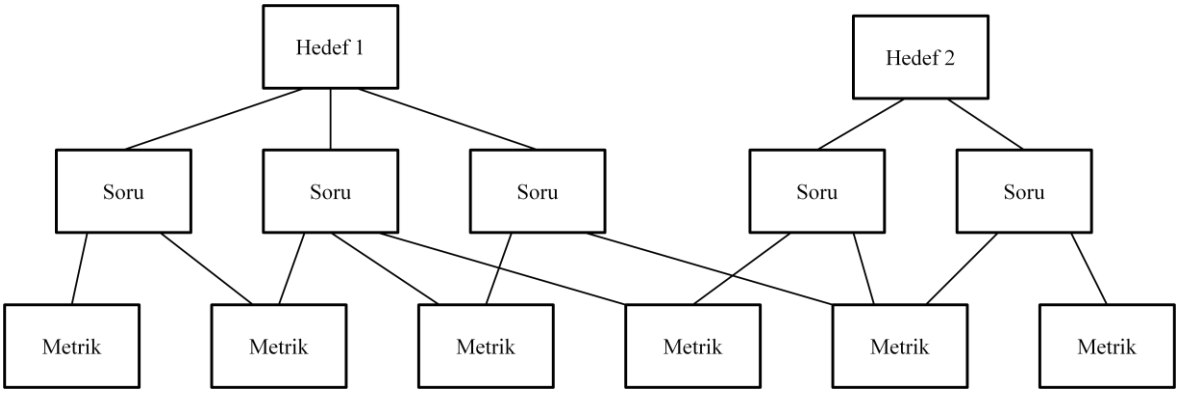
Yazılımda ölçme; gerektirdiği zaman, iş gücü, eğitim, maliyet vb. sebeplerden dolayı, yazılım geliştiren kurumların genelde kaçındığı süreçlerden biridir. Ne var ki ölçme işlemi doğru bir biçimde yapılacak olursa harcanan para ilgili kuruma fazlasıyla geri dönmektedir. Kurumlar ölçme yapacakları alanları belirledikten sonra kullanacakları verileri düzgün bir biçimde kaydedip saklamalıdır. Bu da iyi bir planlama gerektirir. Bir takım kurallar ve yöntemler çerçevesinde gerçekleştirilmeyen veri kaydetme işlemi, sonraki dönemlerde yapılacak olan ölçme ve değerlendirme çabalarını boşa çıkarmaktadır. Bu süreci bir takım kurallar çerçevesinde yürütmek amacıyla Hedef-Soru-Metrik ("Goal-Question-Metric (GQM)") yöntemi ve ISO/IEC 15939-Yazılım Ölçme Süreci standardı gibi kılavuzlar geliştirilmiştir. Ölçme amacının belirlenmesi, planlanması ve gerçekleştirilmesi adımları, bu yöntem ve standartların belirlediği adımlar kullanılarak yapılmalıdır.

Ölçme sürecinin etkili bir şekilde yürütülmesi için [40];

- Kurumsal içeriği, ortamı ve hedefleri anlamak ve karakterize etmek amacıyla uygulanması,
- Ölçme için hedeflerin belirlenmesi ve bu hedefler üzerine yoğunlaşılması,
- Belirlenen ölçme hedefleri doğrultusunda süreçlere, kaynaklara ve ürüne uygulanması gerekmektedir.

Hedef-Soru-Metrik Yöntemi [40], elde edilecek sonuçların kullanım amacına göre ölçme hedeflerini ve metrikleri belirlemeye yarar. Yöntem, çıktıları, bir ağaç şeklinde yapılanan 3 ana adımdan oluşur ve bu adımların gerçekleştirilmesi sonucunda ölçme ve değerlendirmenin sistematik ve kolay bir şekilde yapılacağını

savunur. Bu adımlar; ölçme hedeflerin belirlenmesi, bu hedeflere ulaşıldığını değerlendirmeyi sağlayacak soruların sorulması ve bu sorulara yanıt vermeye yarayacak metriklerin tanımlanmasıdır. Tanımlanan metrikler için toplanan veriler daha sonra, kurulan ağacın yapısına uygun şekilde yorumlanır [40]. Hedef-Soru-Metrik Yöntemi uygulanarak elde edilen yapı, Şekil 2.5'de görüldüğü gibidir; bir metrik, birden çok soruyu yanıtlamaya yarayabilir. Böylece ihtiyaç duyulan veriler analiz edilerek koyulan hedefe ulaşılıp ulaşılmadığı belirlenir.



Şekil 2.5. Hedef-Soru-Metrik Ağaç Yapısı

ISO/IEC 15939 standardı [41] ise yazılım mühendisliği ve yazılım süreç yönetimi alanlarında uygulanmak üzere, yazılım ölçme sürecini tanımlar. İlgili süreç, ölçme faaliyetlerini birbirine bağlayan bir model ile desteklenir. Ölçme faaliyetleri ile ölçme kullanıcısının bilgi ihtiyaçlarına, ölçme ve analiz sonuçlarının nasıl uygulanacağına ve analiz sonuçlarının geçerliliğine nasıl karar verileceğine dair tanımlar üretilir. Ölçme süreci farklı kullanıcıların ihtiyaçlarına göre esnek, değiştirilebilir ve uyarlanabilir olmalıdır. Standart, genel olarak ölçme sürecinin planlanması ve gerçekleştirilmesinde izlenmesi gereken adımları tanımlamaktadır. Ölçme sürecindeki metriklerin seçimi, bilgi ürününün hesaplanması, ölçme sürecinin performansının hesaplanması, bilgi ürünlerinin raporlanması gibi faaliyetlerin nasıl gerçekleştirileceğini de örnekler.

GQM yaklaşımı ve ISO 15939 gibi standartların kullanılmasının yazılım ölçme işlemine büyük katkıları olduğu gibi bir takım dezavantajları da vardır. İlk olarak GQM'e bakacak olursak, GQM yaklaşımı bir ölçme işlemine başlamadan önce yapılması gerekenlerin neler olduğunu belirtmesi ve ölçme yapmaya başlayacak kişilere yol göstermesi nedeniyle büyük faydalar sağlamaktadır. Ölçmenin en önemli aşamalarından biri olan ölçme hedefinin belirlenmesi aşamasına yardımcı

olacak adımlar ve yaklaşımlar sunar. Hedeflerin belirlenmesi ve bu hedefler doğrultusunda nelerin ölçülmesi gerektiğinin belirlenmesi yani bilgi ihtiyacının belirlenmesi ve bu bilgi ihtiyacını karşılayacak yani cevap olacak ölçümlerin, metriklerin belirlenmesi aşamalarının gerçekleştirilmesini sağlayarak ölçme işlemi planlamasının belirli bir standartta gerçekleştirilmesine katkıda bulunur. Ancak, GQM yönteminin de bir takım zorlukları vardır. Bilgi edinmek istediğimiz proje hakkında belirlenecek hedeflerin çıkartılması konusunda çok sıkıntı yaşamasak da, bu hedefler doğrultusunda nelerin ölçülmesi gerektiğinin belirlenmesi için sorulacak soruların hazırlanması aşamasına gelindiğinde, ilgili ölçümü yapacak kişiler bu soruların belirlenmesi aşamasında büyük sıkıntılar çekmektedir. Soruların çok dikkatli ve özenle seçilmesi önemli bir konudur. Bu da büyük bir öngörü gerektirir. Çünkü sorulan sorulara cevap arama aşamasında kestirilemeyen sorunlar çıkabilir ya da ölçme işlemi yanlış yönlendirilebilir. İkinci bir sıkıntı da sorular belirlendikten sonra bu sorulara cevap arama aşamasında kullanılacak metriklerin çıkartılmasına rağmen bu metriklerin nasıl oluşturulacağına dair bir yöntem ya da kurallar dizisi sunmamasıdır. Bu aşamada metriklerin oluşturulma işlemleri ölçüm yapacak kişinin yetenek ve deneyimlerine kalmış bir işlemdir. İlgili metriğin nasıl oluşturulacağı bilinmiyorsa büyük sıkıntılar yaşanabilir.

ISO 15939 standardına bakacak olursak da, bu standart ölçme işlemi esnasında dikkat edilmesi gereken noktaları ve ölçme işleminin belirli standartlarda yapılmasını sağlaması açısından büyük artı sağlamaktadır. Bilgi ihtiyacını çıkarmak ve bilgi ürününü görmek amacıyla sunduğu formlar ölçme işlemini büyük ölçüde kolaylaştırmaktadır. Ancak, ilgili standardın ücretli olması, bu standardın en büyük eksilerinden biridir. Bu standart, ücretli olmasının yanında her şeye kendi bir çözüm yolu önermemesi ve diğer standartlara da sürekli referanslar vererek bir şeyler anlatması büyük sıkıntılar yaşatmaktadır. Diğer standartlara bağımlı olarak anlatımların yapılması, ölçme yapacak kişilerin zaten isteksiz olarak yaptıkları ölçme işini daha da zorlaştırabilir. Özellikle şirketler standartlara kaynak ayırmak konusunda çok da istekli değillerdir. Bu standart, ayrıca ölçme aşamasında karşılaşılabilecek sorunlara tam ve net bir cevap verememekte ya da herkesin bildiği yöntemleri tekrarlamaktadır.

Yazılım geliştirme sürecinin iyileştirilmesi, ilgili kurumların kendi kaynak yönetimlerini ve planlamalarını gerçekleştirebilmeleri açısından ölçme çok önemli

bir yer tutmaktadır. Ölçme işlemi gerçekleştirilmeden kurumların kendileri hakkında bilgi edinmeleri ve yorum da bulunmaları çok da mümkün değildir. Yapılacak ölçümler sırasında kullanılacak verilerin eksikliği, kurumun ihtiyaçlarının tam ve doğru olarak belirlenememesi vb. gibi bir takım sorunlar yaşanabilir. Ölçme işlemi büyük bir vakit ve işgücü gerektirmesi nedeniyle çoğu şirket tarafından tercih edilmemektedir. Çünkü ölçme işleminin yapılabilmesi için geliştirme ekibinin yanında bir de bu işle uğraşan ölçüm ve kalite ekibinin de yer alması gerekmektedir. Ölçme işlemi, sadece bir proje bittikten sonra yapılabilecek bir iş değil bir projeye başlamadan ve proje devam ederken de gerçekleştirilmesi gereken bir iştir. İleriki aşamalarda ölçme yapılabilmesi için projenin yaşam döngüsü boyunca ölçme işlemine ait verilerin de hazırlanması gerekmektedir. Çoğu şirket bunu büyük bir maliyet kaybı olarak görebilir. Kurumlar gerçekleştirecekleri projenin bir an önce bitirilip müşteriye teslim edilmesi için birçok konuda ödünler vermektedir. Bunun için de, bir işin en etkili biçimde yapılması yerine hızlı ve doğru bir sonuç üretilmesi önem kazanmaktadır. Bu da bakım maliyetini artıran etkenlerden biri olmaktadır. Ölçme işlemi için kaybedilen zaman ve işgücü maliyetinden ödün veren şirketler, doğru planlama yapamadıkları için daha çok maliyet ve zaman kaybı yaşamakta, hatta çoğu harcanan emek çöpe gitmektedir. Son olarak elde edilen başka bir deneyim de, ölçme yapacak kişilerin ölçme hakkında bilgi ve deneyim sahibi olmaları gerekmektedir. Bu işlem hem işin hızlı hem de doğru bir biçimde gerçekleştirilebilmesi için çok önemlidir.

2.7. Metrik Kullanılabilirliğinin Değerlendirilmesi

Yazılım sistemlerine ait gerçekleştirilecek kalite değerlendirme ve analizi gibi çalışmalar esnasında, ilgili kalite değerlendirme faaliyetleri uygulanmadan önce faydalanılacak metriklerin kullanılabilirlik açısından değerlendirilmesi büyük bir öneme sahiptir. Kullanılabilirlik açısından uygun görülmeyen yetersiz veya eksik verilerin bu faaliyetler doğrultusunda kullanılması doğru sonuçlar elde etmeyi imkânsızlaştırmaktadır. Uygunluğu değerlendirilmeden kullanılan metriklerin daha sonraki aşamalarda uygun olmadığının fark edilmesi durumu ise o ana kadar harcanan zamanın boşa gittiği anlamına gelmektedir. Bu nedenle kullanılacak metrik ve verilerin uygunluğunun değerlendirilmesi ilk adım olarak uygulanmalıdır.

MKA (Metrik Kullanılabilirlik Anketi - “Metric Usability Questtionnaire”) hem temel hem de türetilmiş süreç metriklerinin nicel analiz için kullanılabilirliğinin

ölçülebilmesi amacıyla bize bir takım sorular içeren anketler sunar [42]. Bu anket soruları ilgili metrik verilerinin varlığını, uygunluğunu ve kullanılabilirliğini anlayabilmek amaçlı oluşturulmuştur. Ayrıca MKA, metrik kullanılabilirliğini ölçen anket sorularına verilen cevapları değerlendirmek amacıyla bir takım derecelendirme kuralları tanımlar.

Metrik kullanılabilirlik niteliklerini derecelendirmek amacıyla dört adet derece("ordinal scale") sunar. Bunlar; Yok("None" (N), Kısmen("Partial" (P)), Büyük Oranda("Large" (L)), ve Tamamen("Fully" (F)). N, en düşük seviyeli dereceye denk gelmektedir. Metrik kullanılabilirlik niteliklerini verilen cevaplara göre değerlendirebilmek ve derecelendirebilmek amacıyla her bir soru için bir beklenen cevap tanımlanmıştır. Bu cevaplar, anket tablolarında beklenen cevaplar başlığı altında en sağdaki kolonda verilecektir.

Ankette yer alan ilk kısım olan "Metrik Kimliği" kısmı, sadece N veya F olarak derecelendirilebilir ve kullanılan metriğin ölçeği Oransal veya Mutlak ise geçerlidir. "Veri Varlığı" kısmı da sadece N veya F olarak derecelendirilebilir ve mevcut veri noktası sayısı 20 üzerinde ise geçerlidir. "Veri Doğrulanabilirliği" kısmı, beklenen cevaplarla uyumlu olarak cevap verilen soruların sayısı açısından değerlendirilir. Bu kısımdaki sorular beklenen cevap sayısına uyumlu soruların sayısının 1, 2, 3 veya 4 olmasına göre N, P, L, F ölçeklerinden biri ile derecelendirilir. "Veri Bağımlılığı", "Veri Normalize Edilebilirliği" ve "Veri Entegre Edilebilirliği" kısımları da "Veri Doğrulanabilirliği" kısmındaki gibi beklenen cevapların girildiği soru sayısının minimum 1, 3, 5 veya 7 olmasına göre N, P, L veya F ölçeklerinden biri ile ölçeklendirilir.

Bir metriğin kullanılabilir olup olmadığına, ankette yer alan sorulara verilen cevaplara göre metrik kullanılabilirlik niteliklerinin derecelendirilmesiyle karar verilir ve metriğin kullanılabilirliği de bu niteliklerin derecesine göre ölçeklendirilir. Bir metriğin kullanılabilirliğine ait dereceler; Güçlü("Strong" (S)), Orta("Moderate" (M)), Zayıf("Weak" (W)) ve Kullanılamaz("None" (N)). N en düşük seviyeli dereceye denk gelmektedir. Metrik kullanılabilirlik niteliklerinin derece sıklıklarına göre bu ölçeklerden birine karar verilir. Nitelikleri 1'den 4'e sıralayacak olursak, eğer bu niteliklere ait ölçekler FFFF veya FFFL ise ilgili metriğin S yani "Güçlü" derecede kullanılabilir olduğunu; FFFP, FFFL veya FFLP ise M yani "Orta" derecede kullanılabilir olduğunu; FFFN, FFLN, FFPP, FFPN veya FFNN ise W yani "Zayıf"

derecede kullanılabilir olduğunu söyleyebiliriz. Bu sıklıklar haricinde kalan diğer tüm sıklıklar ise N yani “Kullanılamaz” derece kullanılabilirliği göstermektedir. MKA, kullanılabilirliği “Güçlü” ve “Orta” seviyesinde olan metriklerin, nicel analizlerde kullanılmasını önermektedir. Daha düşük seviyedeki metriklerle yapılan çalışmalar, güvenilirlik düzeyi düşük çalışmalar olarak kabul edilir.

3. İLİŞKİLİ ÇALIŞMALAR

Her yazılım ürünü geliştirilirken, test edilirken veya bakımı yapılırken farklı teknik, yöntem veya adımlar izlenir. Bu farklılıklar yazılımda ortaya çıkan hataların farklı dağılımlara sahip olması anlamına gelir. Bu sebeple tahminleme için öncelikli olarak üzerinde çalışılacak olan projenin doğasını anlamak çok önemlidir [12]. Bir yazılım sisteminin yaşamı boyunca yazılımda var olan hataların bulunması ve düzeltilmesi süreci, ilgili sistemin karakteristiğini öğrenip anlayabilmek için takip edilebilecek en uygun süreçtir [21]. Böylece sisteme yerleştirilen hatalar aracılığıyla ilgili sistemin nasıl tepkiler verip nasıl davranacağı hakkında fikir sahibi oluruz. Sonrasında ise hataların gösterdiği trende göre geleceğe yönelik hataların göstereceği trendi tahmin etme imkânı buluruz.

Tahminleme modellerinin uygulanması sırasında, modellerin ihtiyaçları yanında bazı ek gereksinimler ortaya çıkmaktadır. Aşağıda, literatürde yer alan çalışmalara ve hata tahminleme konusunda edinilen tecrübelere dayalı olarak bahsedilen ve tahminleme sürecinde karşılaşılan gereksinimler listelenmiştir [11];

Hata tahminleme süreci hedeflerinin proje ve iş hedefleri doğrultusunda sıralanması: Yazılım hata tahminleme yapacak kişi ve kurumlar, kendi proje ihtiyaçları doğrultusunda yürütecekleri tahminleme sürecine özgü hedefleri belirlemelidir. Hata yoğunluğu yüksek olan bileşenler üzerindeki test aktivitelerinin artırılması ya da bakım faaliyetlerine yönelik eforun planlanması, bu hedeflerden bir kaçı olabilir. Hedefler doğru bir şekilde belirlendikten sonra gerçekleştirilecek tahminleme süreci için gerekli planlamalar yapılıp ihtiyaç duyulacak kaynaklar ayrılmalıdır.

Projeye özgü tahminleme modelinin yaratılması: Tahminlemede kullanılacak hata verileri, ilgili projenin geçmiş hata verilerinden toplanmaktadır. Bu nedenle ilgili veriler projeye özgüdür ve geliştirilen tahminleme modeli sadece ilgili proje verilerine uygulandığında doğru sonuçlar elde etmemizi sağlar. Bir çalışmaya [43] göre tahminlemeye yardımcı elemanlar sadece aynı ya da birbirine çok benzeyen projeler için uygulanabilmektedir. Tüm projelere uygulanabilecek bir metrik kümesi bulunamaz.

Sürecin proje ve kurum bazında uygulanabilirliğinin değerlendirilmesi: Gerçekleştirilen tüm tahminleme süreçleri, tahminlenen verinin kalitesinin az

olması [44] ya da verinin yeterli olmaması [45] gibi nedenlerden ötürü başarılı olarak sonuçlanmayabilir. Bu nedenle tahminleme modelinin uygulanabilirliğinin erken aşamalarda değerlendirilip onaylanması, tahminleme hedeflerinin karşılanması aşamasında gereken en önemli durumlardan biridir.

Sonuçların hızlı üretilmesi için duyulan ihtiyacın artması: Tahminleme sürecinin uygulanabilirliğinin başarılı olarak değerlendirilmesi ve onaylanması sürecinden sonra, ilgili tahminleme modelinin hızlı bir şekilde sonuçlar üretmesine ihtiyaç duyulur. Tahminleme süreciyle ilgili literatürde yönlendirici çalışmaların az olması ve tahminlemeyi gerçekleştiren kişilerin yüksek derecede belirsizliklerle karşılaşması nedeniyle elde edilecek sonuçların hızlı bir şekilde üretilmesi zorlaşmaktadır.

Eksik ve yetersiz verilerle uğraşma: Tahminleme sürecinde kullanılacak verilerin toplanması süreci maliyetli ve vakit isteyen bir süreçtir. Bir çalışmada [46] bahsedildiği üzere, eksik veya yetersiz verilerle gerçekleştirilen bir tahminleme süreci sonucu elde edilen sonuçlar, geçerlilik ve doğruluk açısından tehdit altındadır.

Tahminleme sürecindeki belirsizlikler: İlgili tahminleme sürecinde karşılaşılan birçok belirsiz durum, eksik veya yetersiz veri sebebiyle ortaya çıkmaktadır. Bu ise verinin nasıl yorumlanacağı konusunda belirsizliklere yol açmaktadır ve tahminleme sürecini yürüten kişileri bir takım varsayımlar yapmaya zorlar.

Tahminleme modelinin dış kaynaklı yaratılması: Çoğu kurum tahminleme süreci boyunca gereken veri toplama, verinin analiz edilmesi ve tahminleme modelinin kurulması gibi işler için yeterli kaynak ayırmada sıkıntı yaşamaktadır. Kurumların bu gibi tahminleme süreçleri için kaynak ayırması büyük bir lüks olabilir. Bu nedenle ilgili kurumların, tahminleme süreçleri için dışarıdan araç, iş gücü, vb. konularda yardım alması bu süreçte kaybedecekleri zaman ve maliyeti azaltacaktır.

Var olan modelin ileriki sürümlerde kullanılması: Tahminleme modelinin yaratılması esnasında yaşanan zaman ve maliyet kaybını en aza indirgemek için üretilen modellerin tekrar kullanılabilir olması önemli bir durumdur. Ancak, ilgili projeye ait kapsam ve hataların çizdiği dağılım farklılıklar gösterebilir. Bir çalışmada [12] bahsedildiği üzere, yeni hata verileri ortaya çıktıkça ilgili verilerin

var olanlara dâhil edilmesi ve bu nedenle tahminleme modelinin yeniden yaratılması gerekebilir. Tahminleme sonuçlarının doğruluğu için var olan modelin uyarlanması ya da yeni tahminleme modelinin oluşturulması süreci de kurumlara ek maliyetler getirmektedir.

Literatürde birçok tahminleme modeli önerilmektedir. S-Biçimli ve İçbükey-Biçimli ("Concave-Shaped") modeller bunlardan bir kaçıdır. Lineer Regresyon Modeli de literatürde önerilen bu modeller arasında yer almaktadır. Bir çalışmada [47], Lineer Model ile S-Biçimli ve İçbükey-Biçimli modellerin karşılaştırması yapılarak Lineer Regresyon modelinin, diğer yazılım güvenilirlik modelleri kadar iyi performansla tahminleme yapabileceği gösterilmiştir. Başka bir çalışmada [48] ise geliştirilen bir projeye ait planlama, gereksinimlerin belirlenmesi, tasarım, dokümantasyon, geliştirme, test ve bakım gibi aşamalara ait harcanan eforun Lineer Regresyon Modeli kullanılarak geçmiş verilerle tahminlenmesi gerçekleştirilmiştir.

İteratif geliştirme, yazılıma yeni bir özellik katma ya da var olan hataların giderilerek yazılımı daha kaliteli bir duruma getirme amacıyla yazılımı belirli aralıklara sahip iterasyonlar boyunca geliştirmeyi hedefler [49]. Bir çalışmada [22], iteratif olarak geliştirilen projelerde her bir iterasyon sonrasında, sonraki iterasyonlar için tahminleme yapacak dinamik tahminleme modellerinin kurulabileceğinden bahsedilmektedir. Her yeni tahminleme modeli en son gerçekleştirilen iterasyona özgü tahminleme niteliklerinden faydalanılarak kurulabilecek ve daha önceki iterasyonlara ait bilgiler, bu modelin yaratılması sürecinde ek girdi olarak kullanılabilecektir. Bu özelliklerinden dolayı yaratılan her bir tahminleme modeli dinamik bir karakteristiğe sahip olacaktır. Tahminleme değişkenlerinin her bir iterasyonun başında belli olacağını ve bu nedenle her bir iterasyona ait tahminleme modelinin ayrı ayrı kurulması gerektiğini söylemektedir. Bu nedenle, literatürde yer alan tahminleme modellerinin iteratif olarak kullanılması gerektiğinden bahseder. Bir çalışmada [50], iteratif olarak geliştirilen projelerde her bir iterasyonda elde edilen bilgilerin bir sonraki iterasyonlara ait kaliteyi artırıcı etkilere sahip olduğundan bahsetmektedir. Bir iterasyonda ortaya çıkan hatalar dikkate alınarak sonraki iterasyonlarda ortaya çıkabilecek bu tarz hatalardan korunulmasının yazılım kalitesini artırabileceğine değinilmiştir.

Bir önceki bölümde, IEEE 1633 standardı kapsamından bahsedilen güvenilirlik modelleri ile bu tez çalışmasında kullanılan Rayleigh ve Lineer tahminleme

modelleri anlatılmıştır. Rayleigh Modeli kolay uygulanabilirliği, doğru ve güvenilir sonuçlar üretmesi nedeniyle yazılım endüstrisinde oldukça sık tercih edilen modellerden biridir [26]. Literatürde Rayleigh Modeli'nin hataların gösterdiği dağılıma uyarlandığı ve bu modelle tahminlemenin gerçekleştirildiğini örnekleyen çalışmalar mevcuttur [36] [51]. Bu modelden faydalanarak ilgili projelere ait hata sayısı [24], hata yoğunluğu, hata düzeltme eforu ve proje sürümlerinin yayınlanacağı tarihler [26] tahminlenmiştir. İlgili bir çalışma [26], her bir zaman aralığı için bir Rayleigh Modeli'nin kullanılabileceğini ve böylece çoklu Rayleigh Modelleri ile daha doğru tahminleme yapılabileceğini anlatmıştır. Haftalık ortaya çıkan hataların Rayleigh eğrisine uyarlanabileceğinden ve böylece daha sık ve daha doğru tahminlerin üretilebileceğinden bahsetmiştir. Buna yönelik olarak belirlenen zaman aralıklarının her biri için ayrı Rayleigh Modelleri kullanarak iki fazlı bir Rayleigh tahminleme modeli ile öneriyi örneklemiştir.

Yazılım güvenilirliği modelleri yazılımda tahminleme yapmaya olanak veren modellerdir. Bu yazılım güvenilirlik modellerinden NHPP modelini temel alanlar en etkili modeller arasında sayılmaktadır. Bu modeller birçok yazılım [52] [53] projesinde kritik kararların verilebilmesi, maliyet analizi yapılabilmesi, test kaynaklarının ayrılması ve yazılım sürüm tarihi planlaması yapmak amacıyla kullanılmaktadır. Bir çalışmada [49], iteratif olarak geliştirilen bir projeye ait her bir sürümdeki test fazında ortaya çıkan hataların NHPP modeli ile tanımlanması ve ileriye yönelik sürümlere ait hatalarının tahminlenmesi gerçekleştirilmiştir. Bunun için, en etkili NHPP modellerinden biri olan Goel-Okumoto yazılım güvenilirlik modelini [54] kullanmıştır. Başka bir çalışmada [55], NHPP modelleri benzer projelere ait geçmiş hata verilerine doğrultusunda uyarlanarak yazılım güvenilirliğini erken tahminlemeyi sağlayan pratik bir yaklaşım sunulmuştur. Bir diğer çalışmada [56], iteratif olarak geliştirilmiş projelere ait her bir iterasyonun test fazında ortaya çıkan hatalarının regresyon bazlı NHPP modelleri temel alınarak değerlendirilmesi gerçekleştirilmiş ve her bir iterasyonun birbirinden bağımsız ancak istatistiksel olarak benzer olduğu varsayımıyla çoklu lineer regresyon modeli yaklaşımını önermiştir.

COQUALMO modeli literatürde bahsi geçen ve yazılım ürünlerine ait hata yoğunluklarının tahminlenmesi sürecinde son yıllarda tercih edilen modellerden biridir [57] [58]. Ne var ki COQUALMO modeli, üzerinde çalışılacak projeye ait

geliştirme faz bilgilerinin girilmiş olmasını beklemektedir. Bu bilginin eksik olduğu projelerde COQUALMO modeli ile tahminleme yapmak imkânsızlaşmaktadır. Hata yerleştirme ile iteratif olarak geliştirilmiş projelere yönelik hataların davranışını öğrenmeye çalışan ve bunun doğrultusunda hata verilerini geliştirme fazlarına bölerek COQUALMO modeli ile tahminleme gerçekleştiren bir çalışma [21] yer almaktadır. Bu çalışmada hata yerleştirme yaparak ve COQUALMO modelinin yardımı ile hataların erken iterasyonlarda tahminlenebileceğinden bahsedilmiştir.

Bunlar haricinde, regresyon ağaçları ve yapay sinir ağları gibi yöntemleri kullanarak [22] [59] makine öğrenimi [60] [61] aracılığı ile veya Bayesian ağlarını kullanarak [62] projeye ait geliştirme efor tahmini gerçekleştiren çalışmalar da literatürde yer almaktadır.

Bu tez çalışması ile iteratif olarak geliştirilen ve hata faz bilgisi kaydedilmeyen bir proje için tahminleme modeli seçiminin yapılması ve seçilen modellerle gerçekleştirilen tahminleme sonuçlarının değerlendirilmesi hedeflenmiştir. Literatürde; iteratif olarak geliştirilmiş veya yazılım geliştirme faz bilgisinin kaydedilmediği ya da bu verilerin eksik kaydedildiği projeler için hata tahminleme örnekleyen çalışma bulunmamaktadır. Bu çalışmada, hatanın kaydedildiği geliştirme faz bilgisini tutan diğer çalışmalardan [21] [49] [63] [64] farklı olarak hata faz bilgisi kaydedilmediği için bu bilgi dikkate alınmadan tahminleme gerçekleştirilmiştir.

ISO/IEC 9126 standardında [65] belirtildiği gibi iteratif yazılım geliştirme metodolojileri, yazılım kalitesi ve güvenilirliği konuları hakkında fazla pratik sunmamaktadır. Var olan iteratif çalışmalar [22] [21] [47] [49] [56] [63] [64] literatürdeki bu eksikliği gidermeye yönelik gerçekleştirilen çalışmalardır. Bu tez kapsamında da bu konuya yönelik bir katkı sağlanmak hedeflenmiştir. Lineer Regresyon ve Rayleigh Modellerinin iteratif projelerde kullanılabilirliğinin uygunluğu gösterilerek, her projenin kendi Lineer Regresyon ve Rayleigh Model fonksiyonlarını belirleyerek ileriye yönelik tahminleme gerçekleştirilebileceği anlatılmıştır. Ayrıca bu modeller, kolay anlaşılabilir ve fazla matematiksel veya istatistiksel bilgi gerektirmemeleri nedeniyle kolay uygulanabilirliği olan modellerdir. Lineer Regresyon Modeli kullanan çalışmalardaki [22] [47] [56] gibi ve Rayleigh Modeli'ni kullanan çalışmalardaki [24] [26] gibi bu çalışmada Lineer Regresyon ve Rayleigh Modelleri, projenin hata dağılımlarına uygunluğu nedeniyle tercih edilmiş

ve tahminleme sürecinde kullanılmıştır. Bunun yanında, hata verileri, proje ve modül bazında analiz edilerek Lineer Regresyon ve Rayleigh Modelleri kullanılmış ve bir tahminleme çalışması gerçekleştirilmiştir.

Tahminleme modelleri, benzer projelere ait ya da aynı projenin geçmiş hata verilerini kullanarak ileriye yönelik tahminleme gerçekleştirmektedir [24] [26] [56]. Verilerin önceden gösterdikleri bu dağılımları izlemek ileride gerçekleştirecekleri dağılımlar hakkında bizlere fikir vermektedir. İteratif olarak geliştirilen projeler, her bir iterasyonda farklı gereksinimleri karşılamak amacıyla gerçekleştirilseler ve bu da her bir iterasyonda hata dağılımlarını etkileyebilecek faktörlerden biri olsa da, bu tez kapsamında çalışılan projenin bağlamının aynı kalması ve proje ekibinin çok değişkenlik göstermemesi gibi nedenlerden dolayı geçmiş hata verilerinin kullanılması, bize gelecek hata dağılımları hakkında yeterli bilgiyi sağlamaktadır.

Veri toplama sürecinin nasıl gerçekleştirildiğinin, model seçiminin nasıl yapıldığının, tahminleme sürecinde ne tür adımların izlediğinin, bu süreç doğrultusunda karşılaşılan sıkıntı ve zorlukların, öğrenilen derslerin detaylı olarak anlatılması bu çalışmayı diğer çalışmalardan ayıran en önemli özelliklerdir. Bu çalışmada, hem tahminleme yapacak araştırmacılara izleyebilecekleri adımlar önerilerek, hata kaydedilme faz bilgisinin bulunmadığı bir proje üzerinde çalışıp literatürde denenmemiş bir çalışma gerçekleştirilerek hem de karşılaşabilecek zorluklardan ve öğrenilen derslerden bahsedilerek literatüre katkı sağlanmıştır.

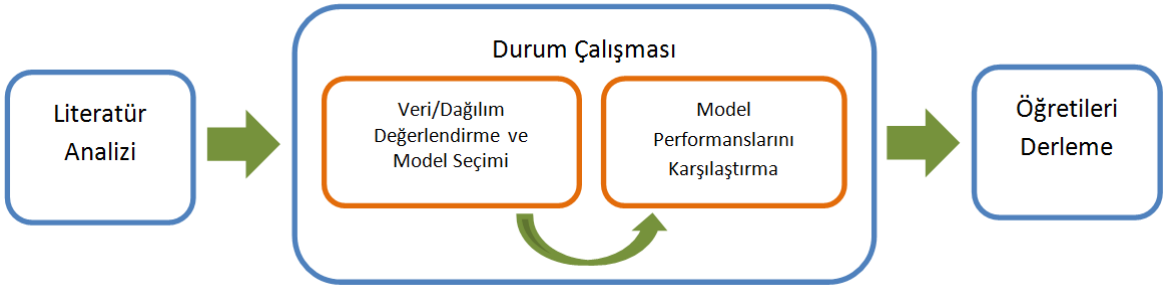
4. YÖNTEM

Bu tez çalışması kapsamında, iteratif olarak geliştirilen ve hata faz bilgilerinin düzenli olarak kaydedilmediği bir projeye ait hata tahminleme süreci tanımlanmış, uygun tahminleme modelleri araştırılmış ve bu modellerle tahminleme gerçekleştirilmiştir. Daha sonrasında ise süreçte karşılaşılan zorluk ve deneyimler derlenmiştir.

İzleyen alt bölümlerde, Giriş bölümünde bahsedilen araştırma soruları doğrultusunda ve yukarıdaki çalışmalara temel olarak, belirlenen araştırma ve çalışma yöntemleri açıklanmaktadır.

4.1. Araştırma Yöntemi

Bu tez çalışmasının geneli için Şekil 4.1'de görülen fazlar planlanmış ve çalışma bu doğrultuda şekillendirilmiştir.



Şekil 4.1. Araştırma Yöntemi Fazları

Bu fazlara detaylı olarak bakacak olursak;

Literatür Analizi: Bu aşama, iteratif olarak geliştirilmiş ve faz bilgisinin kaydedilmediği bir proje için en uygun tahminleme modelinin seçilmesi amacıyla literatür analizinin gerçekleştirildiği aşamadır. Literatürde yer alan tahminleme modelleri incelenerek hangi modelin ne tür çalışmalarda kullanıldığı, kapsamı, ne gibi veri ve bilgilere ihtiyaç duyduğu araştırılmıştır. Öncelikle tahminleme amacıyla kullanılan güvenilirlik modelleri ve istatistiksel modeller, sonrasında ise iteratif olarak geliştirilmiş projelere yönelik tahminleme yapan çalışmalar incelenmiştir. Gerçekleştirilen literatür çalışmasına ait bilgiler, Teknik Bilgiler ve İlgili Çalışmalar bölümlerinde detaylı olarak anlatılmıştır.

Durum Çalışması: Bu aşama, deneysel araştırmanın yapıldığı aşamadır ve şu alt adımlardan oluşmaktadır;

- **Veri/Dağılım Değerlendirme ve Model Seçimi:** Literatür analizi yapıldıktan sonra, hata yoğunluğu verilerine ait değerlendirmeler yapılmış ve kullanılabilirlik analizi gerçekleştirilmiştir. Verilerin uygunluğuna ve gösterdiği dağılımlara göre ve gerçekleştirilen literatür çalışması kapsamında elde edilen bilgiler doğrultusunda tahminleme modellerinin seçimi yapılmıştır. Projeye ait hem modül hem de proje bazında hata yoğunluklarının gösterdiği dağılımlara bakılarak ilgili tahminleme modelleri seçilmiştir. Daha sonra, seçilen modellerin gereksinimleri doğrultusunda, projeye ait hata yoğunluğu verileri kullanılarak tahminleme modelleri kurulmuş ve bu modellerle tahminlemeler gerçekleştirilmiştir.
- **Model Performanslarını Karşılaştırma:** Bu adımda, farklı modellerle gerçekleştirilen tahminleme sonuçlarının değerlendirmesi yapılmıştır. Bunun için kullanılan tahminleme modellerine ait tahminleme performansları hesaplanmıştır. Modellerin performans sonuçları karşılaştırılarak proje ve modül bazında hangi tahminleme modelinin daha iyi olduğu konusuna değerlendirmeler yapılmıştır.

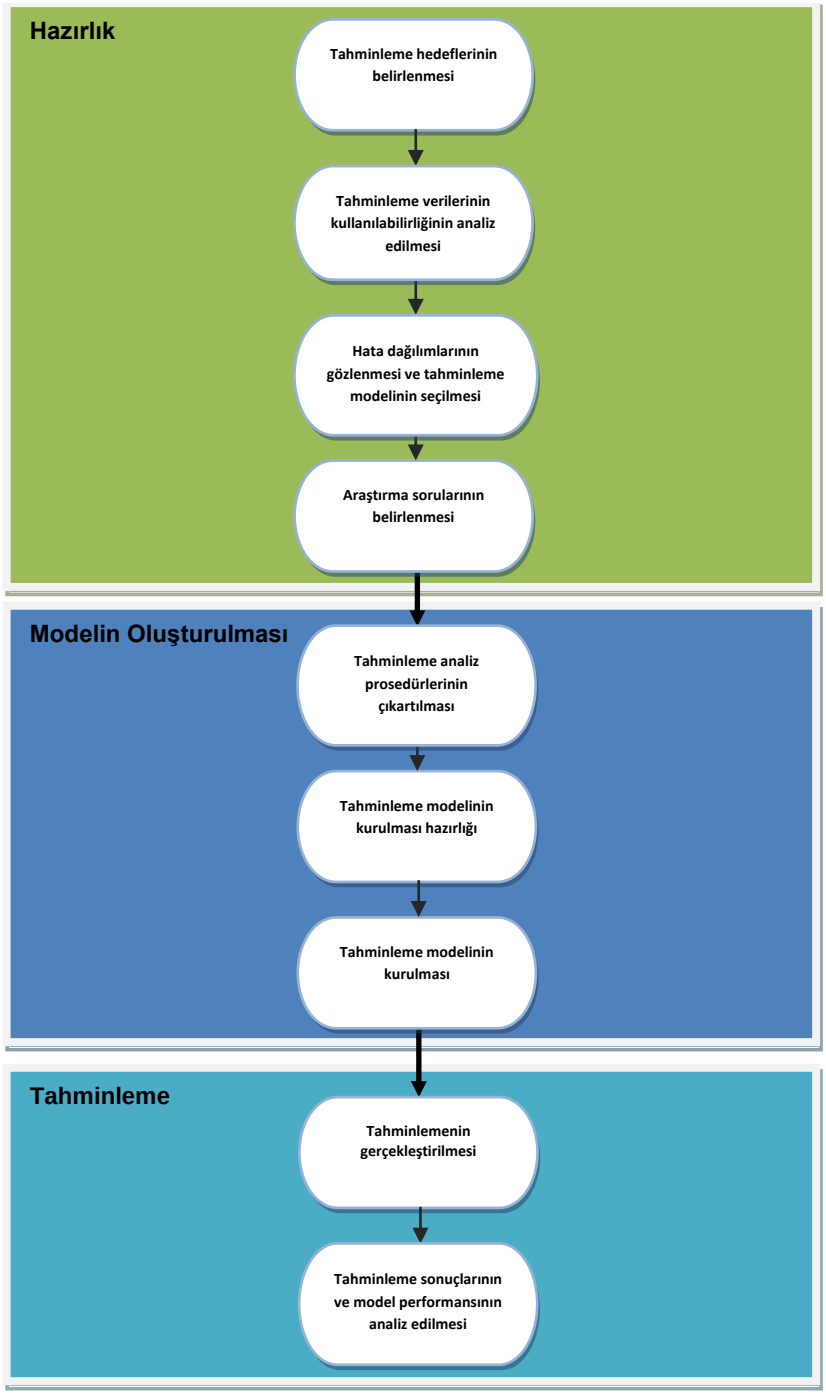
Öğretileri Derleme: Bu son aşamada yürütülen tahminleme süreçlerinin tamamında kazanılan deneyimler, öğrenilen dersler ve karşılaşılan zorluklar değerlendirilerek sonuçlar derlenmiştir. Tahminleme süreci başlatılmadan önce dikkat edilmesi ve göz önünde bulundurulması gereken durumlardan bahsedilmiştir. Bununla, tahminleme süreci yürütecek araştırmacılara, öğretilere dayalı kılavuzluk sağlamak hedeflenmiştir.

4.2. Durum Çalışması Yöntemi

Durum çalışması yürütülürken izlenmesi planlanan adımlar Şekil 4.2'de verilmiş, tahminleme yapacak kişi ve kurumlara örnek oluşturabileceği düşüncesiyle aşağıda açıklanmıştır.

Bir önceki başlıkta verilen "Araştırma Yöntemi" adımları gerçekleştirdiğimiz çalışmanın daha genel hatlarıyla özetini yansıtan bir halidir. Bu adımları detaylandırarak "Durum Çalışması Yöntemi" adımlarına karar verilmiştir. "Araştırma Yöntemi" başlığında yer alan "Durum Çalışması" kısmında verilen "Veri/Dağılım Değerlendirme ve Model Seçimi" adımı, aşağıdaki şekilde verilen "Tahminleme hedeflerinin belirlenmesi", "Tahminleme verilerinin kullanılabilirliğinin

analiz edilmesi", "Hata dağılımlarının gözlenmesi ve tahminleme modelinin seçilmesi", "Araştırma sorularının belirlenmesi" ve "Tahminleme analiz prosedürlerinin çıkartılması" adımlarını kapsamaktadır. "Durum Çalışması" altında verilen bir diğer adım olan "Model Performanslarını Karşılaştırma" adımı ise aşağıdaki şekilde verilen "Tahminleme modelinin kurulması hazırlığı", "Tahminleme modelinin kurulması", "Tahminlemenin gerçekleştirilmesi" ve "Tahminleme sonuçlarının ve model performansının analiz edilmesi" adımlarını kapsamaktadır.



Şekil 4.2. Durum Çalışması Yöntemi ve Adımları

Hazırlık: Bu aşama, tahminleme süreci öncesinde gerçekleştirilen adımları içerir. Bu aşamanın sonunda tahminleme amacının belirlenmiş, tahminlemede kullanılacak verilerin analizinin yapılmış, kullanılacak tahminleme modelinin ve bu modelin nasıl uygulanacağının belirlenmiş olması gerekmektedir. Şu alt adımlardan oluşur;

- **Tahminleme hedeflerinin belirlenmesi:** Bu adımda, gerçekleştirilecek tahminleme sürecinin hangi amaçla yapıldığı, hangi metriklerin kullanılacağı, hangi bilgilerin elde edilmek istendiği belirlenir. Hedef-Soru-Metrik yöntemi [40] bu bilgileri elde etmede kullanılan uygun yöntemlerden biridir.
- **Tahminleme verilerinin kullanılabilirliğinin analiz edilmesi:** Hata sayısı, hata yoğunluğu, hata düzeltme eforu gibi verilerin var olduğunun ve tahminleme için kullanılabilir olduğunun kontrolünün gerçekleştirildiği adımdır. Tahminlemede kullanılacak verilerin yeterli ve eksiksiz olduğunun kontrol edilmesi, tahminleme sürecinin sağlıklı bir şekilde yürütülmesi ve doğru sonuçlar elde edilmesi açısından çok önemlidir. Çalışmada Metrik Kullanılabilirlik Anketi'nden [42] faydalanılarak hata yoğunluğu metriğinin kullanılabilirliği değerlendirilmiştir.
- **Hata dağılımlarının gözlenmesi ve tahminleme modelinin seçilmesi:** Tahminleme için kullanılacak modelin seçiminde faydalanılan en önemli kriterlerden biri, ilgili hataların zaman ekseni üzerinde gösterdiği dağılımın belirlenmesidir. Hataların gösterdiği dağılım ilgili hataların ileride göstereceği davranışlar hakkında ipucu vereceği için, bu dağılıma uygun tahminleme yürütmek, süreci kolaylaştıran etmenlerden biri haline gelmektedir.
- **Araştırma sorularının belirlenmesi:** Bu adımda, tahminleme modellerinin tahminleme sürecinde nasıl kullanılacağı, tahminleme sonucu bu modellerden hangi bilgilerin beklendiğinin belirlenmesi çok önemlidir. Belirlenen araştırma soruları ilgili tahminleme modellerinin kullanımı sürecinin yönetilmesinde etkin rol oynar. Belirlenen sorular tahminleme hedeflerinin gerçekleştirilmesine hizmet eder.

Modelin Oluşturulması: Bu aşama, hazırlık aşamasında belirlenen tahminleme hedefleri doğrultusunda ve seçilen tahminleme modellerinin gereklerine göre, amaca özel tahminleme modellerinin kurulması adımlarını içerir. İlgili tahminleme modellerine ait parametreler ve katsayılar hesaplanır ve tahminleme modellerine ait fonksiyonlar elde edilir. Şu alt adımları içerir;

- **Tahminleme analiz prosedürlerinin çıkartılması:** Bu adım seçilen tahminleme modellerinin nasıl oluşturulacağını anlatan detaylı adımların

tanımlandığı kısımdır. Modelin nasıl kullanılacağı, parametrelerin nasıl hesaplanacağı ve hangi verilerin modeli eğitmek için kullanılacağı bu adımda belirlenir. Seçilen modele göre tahminleme sürecini yönetecek kurumlar, bu analiz prosedürlerini kendi hedefleri doğrultusunda şekillendirebilirler.

- **Tahminleme modelinin kurulması hazırlığı:** Model parametrelerinin hesaplanabilmesi için hangi verilerin kullanılacağı ve modelin nasıl eğitileceğini anlatan adımdır. Bir projeye ait geçmiş veriler olabileceği gibi aynı kuruma ait benzer başka proje verileri de bu amaçlı kullanılabilir.
- **Tahminleme modelinin kurulması:** Belirlenmiş tahminleme modellerine ait tahminleme fonksiyonlarının ve bu fonksiyonlara ait katsayıların belirlendiği adımdır. Tahminleme fonksiyonu, ilgili parametreler yerine konarak her bir veri noktasında ilgili tahminleme sonuçları elde edilecek şekilde hazır hale gelir.

Tahminleme: Bu aşama, kurulan tahminleme modelleri ile tahminlemenin gerçekleştirildiği ve tahminleme sonuçlarının elde edildiği adımdır. Elde edilen tahminleme sonuçlarının analizi de gerçekleştirilir. Şu alt adımlardan oluşur;

- **Tahminlemenin gerçekleştirilmesi:** Bu adım, elde edilen tahminleme fonksiyonlarında ilgili parametreler yerine konarak her bir veri noktasında ilgili tahminleme sonuçlarının hesaplandığı adımdır.
- **Tahminleme sonuçlarının ve model performansının analiz edilmesi:** Tahminleme sonucu elde edilen değerlerin gerçek değerlerle karşılaştırıldığı ve tahminleme modellerinin ne derecede doğru sonuçlar ürettiğinin kontrolünün yapıldığı adımdır. Bir takım metrikler aracılığıyla modellerin ürettiği sonuçlar değerlendirilerek tahminleme modellerinin performansları hesaplanır. Birden çok tahminleme modelinin kullanıldığı durumlarda ilgili modellerin performansları karşılaştırılarak değerlendirmeler yapılır.

Tahminleme sürecini uygulayacak kişi ve kurumlar bu adımları izleyerek ve kendi ihtiyaçları doğrultusunda detaylandırarak tahminleme süreçlerini yönetebilirler. Takip eden bölümde bu adımların durum çalışmasında nasıl gerçekleştirildiği anlatılmıştır.

5. DURUM ÇALIŞMASI

Bu bölümde, tez çalışması kapsamında gerçekleştirilen durum çalışması ve çalışma sonuçları detaylı olarak açıklanmıştır.

Çalışma kapsamında, bir önceki bölümde bahsedilen yöntem ve teknikler kullanılarak bir projeye ait hata verileri incelenmiş ve uygun tahminleme modelleri seçilerek ilgili modeller kurulmuştur. Kurulan bu modeller ile ilgili projeye ait ileriye dönük tahminleme gerçekleştirilmiştir. İlgili işlemler sonrası hata tahminleme sonuçlarının gerçeğe yakınlığı analiz edilmiştir. Bu kapsamda, hata verilerinden faydalanılacak olan projenin kapsamı hakkında bilgiler edinmek ve ilgili projeyi yakından tanımak, tahminleme sürecinin yürütülmesinde faydalı olmuştur.

5.1. Çalışma Bağlamı

Üzerinde çalışılan proje iteratif olarak geliştirilen bir kamu projesidir. Yaklaşık üç yılın üzerinde bir süredir geliştirilmekte olup proje hala devam etmektedir. İlgili projeye ait şu ana kadar 14'e yakın modül geliştirilmiştir ve yeni modüller geliştirilmeye devam etmektedir. Projede 30'a yakın kişi çalışmaktadır ve bu kişiler, en az bir modülle ilgilenen 4 ila 7'şer kişilik olmak üzere çeşitli ekiplere bölünmüştür. İlgili modüller kendi içerisinde alt bileşenlerden meydana gelir. Proje ekibinde yer alan her bir geliştirme ekibi üyesi, bu bileşenlerden en az bir ya da birkaçından sorumludur. Projede çalışan ekip, kişi sayısı ve çalışan kişiler, çok fazla değişikliğe uğramamıştır. Proje; bir proje yöneticisi, bir proje yönetici yardımcısı, geliştirme ekipleri ve bu ekiplerin takım liderleri, bir konfigürasyon yöneticisi ve eğitim ekibinden oluşmaktadır.

Geliştirme ekibi, müşteriden gelen talepler doğrultusunda yazılımı geliştirme, test etme, bakımını yapma gibi faaliyetlerden sorumludur. Projede tüm geliştiriciler, yazılımın test edilmesinden sorumludur. Bu kapsamda geliştiriciler yazılımın test edilmesi amacıyla birim testlerin, entegrasyon testlerinin, ve otomatik testlerin geliştirilmesini gerçekleştirmektedir. Projenin konfigürasyon yöneticisi, her bir sürüm öncesi test ortamlarını ve test veri tabanlarını hazırlar. Daha sonra, bir sürüm alınmadan önce tüm testlerin koşturulması sağlanır ve bu testlerin başarılı bir şekilde geçmesi için geliştiriciler tarafından düzeltme faaliyetleri gerçekleştirilir. Elle işletilen testler harici testlerin tamamı, devamlı entegrasyon aracı("Jenkins")

[66] tarafından her gece koşturulmaktadır. Testleri geçmeyen geliştiricilere ilgili araç tarafından otomatik olarak bilgilendirici e-posta atılır.

Bunlar haricinde geliştirme ekibi, müşteriden gelen gereksinimlerin yazılması ve bu gereksinimlerin bakımının yapılması sürecinden de sorumludur. Geliştirme faaliyetleri sonrasında müşteri ile toplantılar yapılarak elle işletilen ve otomatik olarak işletilen testler çalıştırılır ve geliştirilen gereksinimlerin geçerlenmesi sağlanır. Ayrıca, geliştirme ekibi tarafından gerçekleştirilen tüm talepler, ilgili geliştiricinin üyesi olduğu ekibin başka bir üyesi tarafından doğrulanarak ilgili talep kapatılır. Eğer, talep doğrulanmazsa başarısız olarak kabul edilir ve düzeltmesi için ilgili geliştiriciye tekrar yönlendirilir.

Projede, geliştirme ekipleri arasında yer alan bir adet altyapı ekibi bulunmaktadır. İlgili ekip, projede kullanılması öngörülen ve geliştiricilerin hızlı ve etkin kod yazmalarını sağlayacak bileşenlerin geliştirilmesinden sorumludur. Bu ekip ayrıca, sistem performansının artırılmasına yönelik devamlı olarak çalışma gerçekleştirmektedir.

Eğitim ekibi, ilgili projeye ait yazılım ürününün son kullanıcılarına, ürünün nasıl kullanılması gerektiğine dair belirli sürelerle eğitimler vermekle sorumludur. Ayrıca, müşterilerin kullanım esnasında ihtiyaç duyduğu bilgiye anında cevap vererek kullanıcıların sistemi rahat ve etkin kullanmalarını sağlamaktadır. Bunun için eğitim ekibi, kullanım kılavuzu yazarak ve gerekli güncellemeleri yaparak müşteriye sunmaktadır.

Projeye ait tüm paydaşlar tarafından oluşturulan talepler, bir talep yönetim sistemi (“JIRA”) [67] tarafından yönetilmektedir. Bu talep yönetim sistemi, tüm paydaşların erişimine açıktır ve paydaşların taleplerini ilgili kişilere iletmeleri, bu şekilde sağlanmaktadır. Proje paydaşları; geliştirme, gereksinim değişikliği, gözden geçirme, hata bildirme vb. türlerde talepler oluşturabilmektedirler. Oluşturulan bu talepler, hangi bileşeni ilgilendiriyorsa ilgili bileşenden sorumlu kişiye, bu talebi çözmesi için takım lideri tarafından atanmaktadır. İlgili talepler açıldıkları andan itibaren bir dizi durum değişikliklerine uğrarlar. Bu durum değişiklikleri; ilgili talebin açılması, kabul edilmesi, çözüm sürecinde olması, çözülmesi ve doğrulama sonucu kapatılması gibi durumları içerir. Bu durum değişiklikleri ilgili talep yönetim sistemi tarafından talebin geçmişinde değişiklik tarihleri ile birlikte tutulmaktadır.

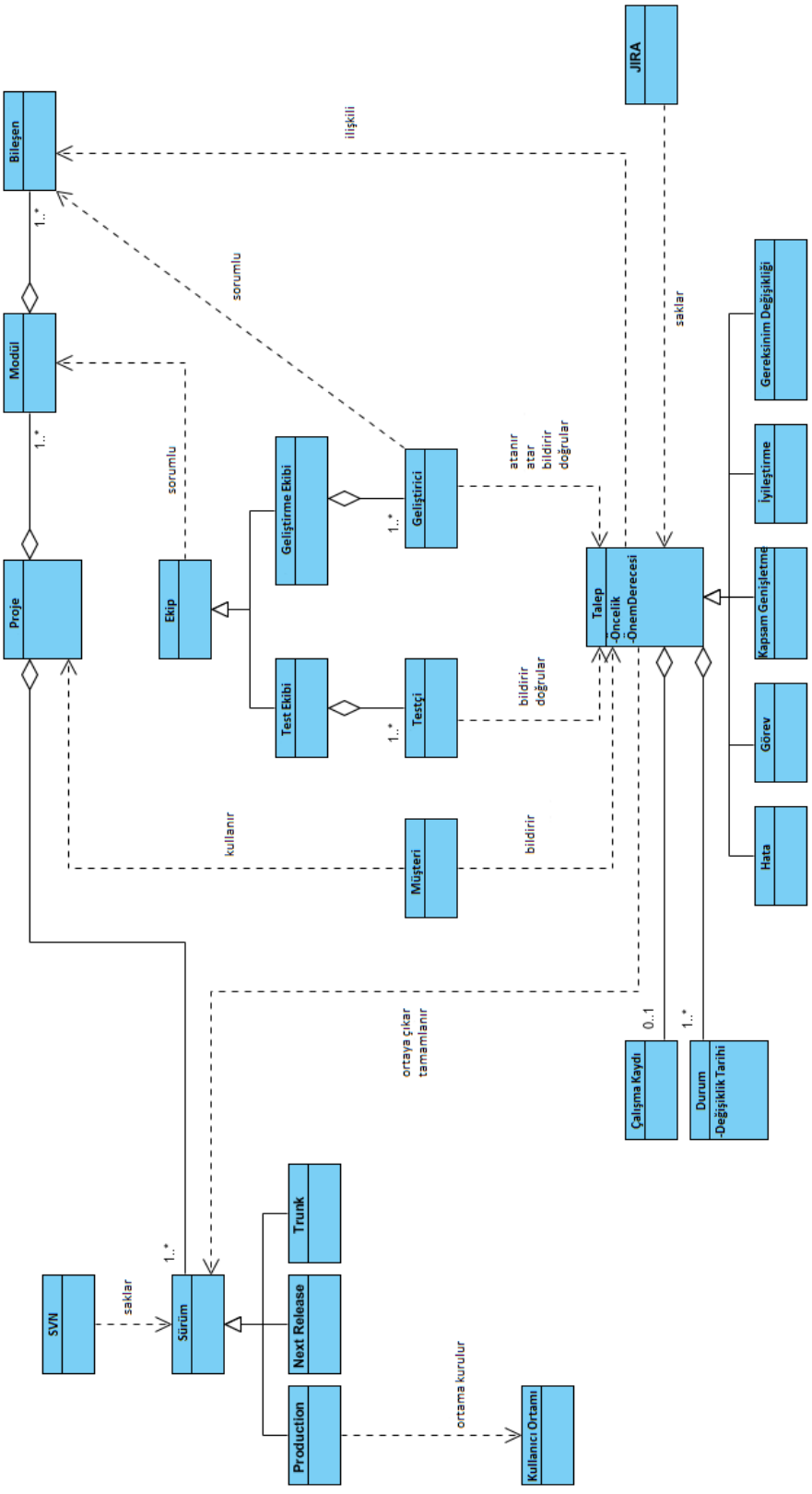
Ayrıca, ilgili talebin çözülmesi aşamasında değiştirilen kodlara ait bilgiler de talep yönetim sistemi tarafından tutulmaktadır. Talepler atanan kişi tarafından ne kadar çalışıldığı bilgisini ("worklog") saklayabilme özelliğine sahiptir. Ancak, bu çalışma süresi bilgisi, ilgili geliştiricinin inisiyatifine bırakılmış ve girilmesi zorunlu olmayan bir alandır. Talep yönetim sisteminde oluşturulan taleplerde en önemli bilgi eksikliği, ilgili talebin yazılım yaşam döngüsünün hangi fazına (gereksinim analizi, tasarim, kodlama vb.) ait olduğunun kaydedilmemesidir. Gelen talebin yaşam döngüsünü hangi fazını ilgilendirdiğinin bilinmemesi, faza dayalı tahminleme yapan modellerin kullanımını kısıtlamaktadır.

Projenin sürümlerine ait kaynak kodları, SVN("Subversion") [68] konfigürasyon aracı tarafından saklanmaktadır. Devamlı entegrasyon ortamı, SVN, test ortamlarının kurulması gibi süreçler projedeki konfigürasyon yöneticisi tarafından yürütülmektedir. Konfigürasyon yöneticisi ayrıca, müşteri ortamına ürünün yüklenmesinden ve sürümler arası geçişlerde gerçekleştirilecek entegrasyon işlemlerinden sorumludur.

Proje, birbirini takip eden üç çeşit sürüm halinde iteratif olarak geliştirilmektedir. Bunlar, ana geliştirmenin gerçekleştirildiği yani yazılıma yeni gereksinimlerin ve özelliklerin eklendiği ya da iyileştirmelerin yapıldığı "Trunk" sürümü; "Trunk" sürümünde yapılan değişikliklerin test edilmesi ve test sonuçlarına göre hataların düzeltilmesi amacıyla alınan "Next Release" sürümü ve "Next Release" sürümü testlerden geçtikten sonra alınan ve kullanıcı ortamına yüklenen, kullanıcıdan gelen hataların düzeltildiği "Production" sürümüdür. "Trunk" ana sürüm olarak kabul edilir ve yaklaşık 4-6 hafta arası bir geliştirme süresine sahiptir. Her bir ana sürüm ("Trunk") bir iterasyona denk gelmektedir. Bir sürüme ait Trunk sürümü geliştirilirken eş zamanlı olarak "Production" sürümünden gelen müşteri kaynaklı hatalar düzeltilmektedir. Geliştirme ekibi, müşteri hatalarına öncelik verecek şekilde çalışır. "Next-Release" sürümü, "Trunk" sürümünde gerçekleştirilen geliştirme faaliyetleri sonlandıktan sonra alınmaktadır ve daha çok yeni geliştirilen kodların test edilmesi amacıyla alınır. Test edilip düzeltilen hatalar sonrası bir-iki adet daha alınan "Next-Release" sürümleri sonucu testlerde fark edilen hatalar düzeltilmiş ve "Production" sürümü alınmaya hazır duruma gelinmiş olur. "Production" sürümü alındıktan sonra ilgili sürüm müşteri ortamına yüklenir ve müşterinin kullanımına açılır. Burada müşteriden gelen hatalar biriktirilip bir-iki

günlük düzeltme faaliyetleri sonucu yeni “Production” sürümleri alınarak müşteri ortamına yeni kurulumların yapılmasına devam edilir.

Şekil 5.1’deki kavramsal model, yukarıda bahsedilen proje kapsamındaki varlıkları ve ilişkilerini özetlemektedir.



Şekil 5.1. Proje Bağlamı Kavramsal Modeli

5.2. Durum Çalışması

Bu kısımda, yukarıda bağlamı hakkında bilgi verilen, iteratif olarak geliştirilmiş, hataların kaydedildiği faz bilgisinin bulunmadığı bir kamu projesi kapsamında geliştirilen yazılım ürününe ait hata yoğunluğunun tahmin edilmesi sürecine yönelik adımlar anlatılmıştır. İlgili adımlar, durum çalışmasına ait verilen yöntem doğrultusunda detaylandırılmıştır.

5.2.1. Hazırlık

Bu aşamada tahminleme süreci öncesi gerçekleştirilen adımlardan bahsedilecektir.

5.2.1.1. Tahminleme hedeflerinin belirlenmesi

Bu çalışma kapsamında ilgili yazılım sisteminde ortaya çıkan hataların nasıl bir dağılım sergilediğini anlayabilmek ve hataların gösterdiği bu dağılıma göre ileriye yönelik tahminleme sürecini planlayabilmek amacıyla ölçüm hedeflerinin ve ölçülecek niteliklerin belirlenebilmesi için analiz çalışması yapılmıştır. Bu amaçla Hedef-Soru-Metrik [40] yönteminden faydalanılmıştır. Yapılacak ölçüm işlemi için belirlenen hedef, sorular ve metrikler şöyledir:

Hedef: Projeye ait hatalar ve hata dağılımı hakkında bilgi edinmek.

Bu hedef için tanımlanan sorular;

Soru 1: Yazılımın her bir ana sürümdeki hata yoğunluğu nedir?

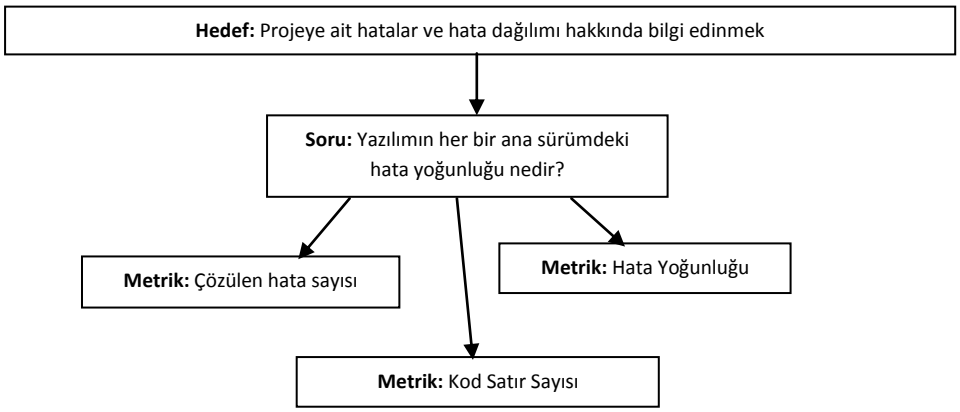
Bu soruları cevaplamak için gerekli bilgiler(metrikler);

Metrik 1: İlgili sürüm boyunca çözülen hata sayısı

Metrik 2: Yazılım ürününün, ilgili sürümdeki, kod satır sayısı cinsinden büyüklüğü

Metrik 3: Hata yoğunluğu(**Hata Sayısı / Kod Satır Sayısı**)

Bu yapıyı Hedef-Soru-Metrik ağacında ifade edecek olursak;



Şekil 5.2. Çalışmaya ait Hedef-Soru-Metrik Ağacı

İlgili yazılıma ait hata dağılımı bilgisini gözlemleyebilmek için bir takım bilgilere ihtiyaç duymaktayız. Burada ihtiyaç duyduğumuz ilk bilgi, belirli zaman aralıklarında(sürüm zamanları ya da proje yaşamının eşit aralıklara bölünmüş hali de olabilir), ilgili yazılım sürümüne ait çözülen hata sayısının toplam ürün büyüklüğüne oranı yani yazılımın hata yoğunluğudur. Her bir zaman diliminde ölçülen hata yoğunluğu bilgisi, yazılımın yaşamı boyunca hata yoğunluğu dağılımı ve yazılımın ileriye dönük göstereceği dağılım hakkında bize bilgi sağlamaktadır. Bu dağılımın göstereceği örüntüye göre uygun model seçimi yapılarak ileriye dönük hatalılık tahmini yapılabilmektedir.

Aşağıda belirlediğimiz bilgi ihtiyaçları doğrultusunda hazırlanmış Ölçme Yapısı [41] çizelgesi yer almaktadır. Bu çizelgede belirlenen bilgi ihtiyaçlarını karşılamak için ihtiyacımız olan bilgiler, detaylı bir şekilde bulunmaktadır. Yaklaşık birer aylık periyotlar halinde ana sürümler alınması nedeniyle, üzerinde çalıştığımız yazılım ürününün zaman birimini, sürüm sayısı cinsinden ölçerek sürecimize dâhil etmekteyiz. Çizelge 5.1'de Hedef-Soru-Metrik modelimize ait "Yazılımın her bir ana sürümdeki hata yoğunluğu nedir?" bilgi ihtiyacı için oluşturulmuş Ölçme Yapısı çizelgesi verilmiştir.

Çizelge 5.1. Yazılımın Her Bir Ana Sürümündeki Hata Yoğunluğu Bilgi İhtiyacı Ölçme Yapısı

Bilgi İhtiyacı	Yazılımın her bir ana sürümündeki hata yoğunluğu hakkında bilgi edinmek
Ölçülen Kavram	Hata yoğunluğunun zamana göre dağılımı
İlişkili Varlıklar	1. Yazılım her bir ana sürümünde ortaya çıkan hatalar 2. Yazılım her bir ana sürümünün kaynak kodu
Nitelikler	1. Hataların sayısı 2. Yazılımın büyüklüğü
Temel Metrikler	1. Çözülen hata sayısı 2. Yazılım kod satır sayısı
Ölçme Yöntemi	1. Sürümde çözülen hata sayısının sayılması 2. Yazılımdaki kaynak kodda boşluk ve yorum satırları haricindeki satırların sayılması
Ölçme Yöntemi Türü	1. Objektif 2. Objektif
Ölçek	1. Sıfırdan sonsuza sayılar 2. Sıfırdan sonsuza sayılar
Ölçek Türü	1. Aralık 2. Aralık
Ölçme Birimi	1. Hata Sayısı 2. Satır Sayısı
Türemiş Metrik	Hata Yoğunluğu
Ölçme Fonksiyonu	Hata sayısının kod satır sayısı olarak yazılım büyüklüğüne bölünmesi
Gösterge	Zaman-Hata Yoğunluğu dağılım grafiği
Model	Dağılımın güvenilirlik modelleri dağılımlarıyla karşılaştırılması
Karar Kriterleri	Bulunan dağılımın hangi güvenilirlik modeli dağılımına benzediği kontrol edilecek.

Yukarıda verilen Ölçme Yapısı tablosu ile elde edilmek istenen ve analiz edilecek olan bilgi ürünlerinin tasarımı gerçekleştirilmiştir.

Ölçme sürecinde kullanacağımız metriklerden bahsedecek olursak; ilk olarak yazılımın ilgili sürümüne ait hata yoğunluğunun dağılımını görebilmek için, “hata sayısı” ve “kod satır sayısı” metriklerinden türeyen “hata yoğunluğu” metriğine ihtiyaç duymaktayız. Bu türetilmiş metriği elde edebilmek için yazılımı, her biri yaklaşık 4-6 haftalık periyotlar halinde süren ana sürümlere bölerek her bir sürümde kaç adet hatanın çözüldüğü bilgisine (temel metrik) ve ilgili sürümün kod

satır sayısı bilgisine (temel metrik) ihtiyaç duyulmaktadır. “Kod satır sayısı” metriği her bir sürümde ortaya çıkan hataları normalize edebilmek amacıyla hesaplanmaktadır. Bulunan hata sayıları bu büyüklük metriğine bölününce “hata yoğunluğu” türetilmiş metriğini elde etmiş oluyoruz.

Yazılıma ait hata verileri, ilgili kurum tarafından kullanılan yazılım talep yönetim sisteminde(“JIRA”) [67] saklanmaktadır. Bu veriler, talep yönetim sisteminin görsel ara yüzünden, talep türü hata seçilerek ve sürüm tabanlı sorgulama yapılarak elde edilmiştir. Sorgulanan hata verileri sürümler halinde gruplu bir biçimde Excel dosyasına çıktı olarak alınmıştır. Yazılımın büyüklüğünü elde edebilmek içinse toplam kod satır sayısı bilgisi kullanılmaktadır. Her bir sürüme ait toplam kod satır sayısı bilgisini hesaplayabilmek için, sürümlere ait kaynak kodlar, SVN [68] konfigürasyon aracından çekilmiştir. Daha sonra açık kaynak kodlu bir kod satır sayısı hesaplama uygulamasından faydalanılarak her bir sürüme ait toplam kod satır sayısı bilgisi elde edilmiştir.

Yukarıda tanımlanan her bir bilgi ihtiyacı için kullanılacak metrikler ve ölçülen veriler analiz edilerek bu verilerin yeterliliğinin ve kullanılabilirliğinin kontrolü yapılmıştır. Bunun için metrik kullanılabilirliğini değerlendirmemizi sağlayan MKA [42] formları kullanılmıştır.

5.2.1.2. Tahminleme verilerinin kullanılabilirliğinin analiz edilmesi

İlgili projeye ait hata yoğunluğunun tahminlenmesi süreci öncesinde, tahminleme boyunca gereken metriklerin kullanılabilirliği ve yeterliliği açısından değerlendirilmesi gerekmektedir. Çünkü kullanılacak metriklere ait eksik veya yetersiz verilerin bulunması, oluşturulacak tahminleme modelinin doğru sonuçlar üretmesine engel oluşturur.

Tahminleme için kullandığımız hata yoğunluğu metriği; hata sayısı ve kaynak kodun satır sayısı cinsinden büyüklüğü temel metriklerinden hesaplanmış bir türetilmiş metriktir. Bu nedenle hata yoğunluğu metriğinin kullanılabilirliğinin değerlendirilmesi işleminden önce, bu metriği oluşturan hata sayısı ve kod büyüklüğü metriklerinin kullanılabilirliğinin değerlendirilmesi gerekmektedir. Bu iki temel metriğin kullanılabilir olmadığına karar verilirse hata yoğunluğu türetilmiş metriğinin kullanılması da anlamsızlaşmaktadır. Kullanılacak tüm bu metriklerin

kullanılabilirliğinin değerlendirilmesi amacıyla MKA'nın sunduğu formlardan faydalanılmıştır [42].

Aşağıda hata sayısı, kod satır büyüklüğü ve hata yoğunluğu metriklerine ait kullanılabilirlik anketlerinin, üzerinde çalışılan projeye ait veriler göz önüne alınarak doldurulmuş halleri verilmiştir.

Hata sayısı: Herhangi bir hata fark edildiğinde ilgili hata talebi sisteme girilmektedir. Ayrıca, hata sisteme girildiği anda hangi sürümü etkilediği, hangi bileşenden kaynaklandığı gibi bilgiler de elimizde mevcuttur. Böylece bir sürümde kaç adet hatanın açıldığı, çözüldüğü gibi bilgileri elde edebiliriz.

Çizelge 5.2. Hata Sayısı Kullanılabilirlik Anketi

			Lütfen her bir niteliği, sorulara verilen cevaplar doğrultusunda aşağıda verilen dört ölçekle derecelendiriniz:		
Metrik Adı: Hata Sayısı			F : Niteliğe ait göstergeler tamamen sağlanıyor (%86-100)		
Kavramsal Tanım: Tüm paydaşlar tarafından sisteme kaydedilen hataların miktarıdır.			L : Niteliğe ait göstergeler büyük oranda sağlanıyor (%51-85)		
Değerlendirme Tarihi : 01/03/2014			P : Niteliğe ait göstergeler kısmen sağlanıyor (%16-50)		
Değerlendiren: Anıl AYDIN			N : Niteliğe ait göstergeler yok ve sağlanamıyor (%0-15)		
Nitelikler			**Cevaplar**	**Derece**	Beklenen Cevaplar
	Göstergeler				
Metrik Kimliği			**MKF-1**	**F**	
	S1	Metrik, hangi varlığı ölçüyor?	Hata		
	S2	Metrik varlığın hangi niteliğini ölçüyor?	Hata Miktarı		
	S3	Metrik verisinin ölçeği nedir? (nominal, sıralı, aralıklı, oransal, mutlak)	Oransal		Oransal, Mutlak
	S4	Metrik verisinin birimi nedir?	Adet		
	S5	Metrik verisinin türü nedir? (integer, real, etc.)	Sayı		
	S6	Metrik verisinin aralığı nedir?	Proje başından itibaren hata miktarı		
Veri Varlığı			**MKF-2**	**F**	
	S7	Metrik verisi mevcut mu?	Evet		Mevcut > 20
	S8	Tüm gözlemlerin miktarı nedir?	Farkedilen tüm hatalar kaydediliyor. Yaklaşık 7000 adet hata		
	S9	Eksik veri noktalarının miktarı nedir?	Yok, hatalar eksiksiz kaydediliyor.		
	S10	Veri noktaları belirli periyodlar halinde mi eksik? (Evet ise, gözlemlenen bu periyodların sayılarını belirtiniz)	Hayır		
	S11	Metrik verisi zaman sıralı mıdır? (Hayır ise, metrik verisinin nasıl sıralandığını belirtiniz)	Evet, metrik verisi zaman sıralıdır.		
Veri Doğrulanabilirliği			**MKF-3**	**F**	
	S12	Metrik verisi sürecin hangi aşamasında kaydedilmiştir? (başlangıçta, ortada, sonda, daha sonra, vb.)	Hatalar farkedildikçe sürecin her aşamasında kaydediliyor. (Hata düzeltme faaliyetlerinin başında)		
	S13	Tüm metrik verileri sürecin aynı aşamasında mı kaydedildi? (başlangıçta, ortada, sonda, daha sonra, vb.)	Evet		Evet
	S14	Metrik verisini kaydetmeden kim sorumludur?	Tüm paydaşlar		
	S15	Tüm metrik verisi sorumlu kişi tarafından mı kaydedildi?	Evet		Evet
	S16	Metrik verisi nasıl kaydedildi? (form ile, rapor ile, araç ile, vb.)	Talep yönetim sistemi(JIRA) üzerinden		
	S17	Tüm metrik verileri aynı şekilde mi kaydedildi? (form ile, rapor ile, araç ile, vb.)	Evet		Evet
	S18	Metrik verileri nerede saklanıyor? (bir dosyada, veritabanında, vb.)	Talep yönetim sistemi(JIRA) veri tabanında		
	S19	Tüm metrik verileri aynı yerde mi saklanıyor? (bir dosyada, veritabanında, vb.)	Evet		Evet
Veri Bağımlılığı			**MKF-4**	**F**	
	S20	Metrik verisinin yaratılma sıklığı nedir? (asenkron, günlük, haftalık, aylık, vb.)	Asenkron		
	S21	Metrik verisinin kaydedilme sıklığı nedir? (asenkron, günlük, haftalık, aylık, vb.)	Asenkron		
	S22	Metrik verisinin saklanma sıklığı nedir? (asenkron, günlük, haftalık, aylık, vb.)	Asenkron		
	S23	Verilerin yaratılma, kaydedilme ve saklanma sıklıkları farklı mı?	Hayır		Hayır
	S24	Metrik verisi eksiksiz olarak kaydediliyor mu?	Evet		Evet
	S25	Metrik verisi belirli bir amaç için mi toplanıyor?	Evet		Evet
	S26	Metrik verisinin toplanma amacı süreci yürüten kişiler tarafından biliniyor mu?	Evet		Evet
	S27	Metrik verisi analiz edilip raporlanıyor mu?	Evet		Evet
	S28	Metrik verisi analiz sonuçları süreci yürüten kişilerle paylaşılıyor mu?	Evet		Evet
	S29	Metrik verisi analiz sonuçları süreci yönetimle paylaşılıyor mu?	Evet		Evet
	S30	Metrik verisi analiz sonuçları karar verme amaçlı kullanılıyor mu?	Evet		Evet
Veri Normalize Edilebilirliği					
	S31	Metrik verisi parametreler veya metriklerle normalize edilebilir mi? (Evet ise, hangileri belirtiniz)	Evet, kod büyüklüğü		
Veri Entegre Edilebilirliği					
	S32	Metrik verisi proje düzeyinde entegre edilebilir durumda mıdır?	Evet		
	S33	Metrik verisi kurumsal düzeyde entegre edilebilir durumda mıdır?	Evet		

Yukarıda doldurulan anket sonuçlarını değerlendirecek olursak; doldurulan değerler beklenen anket cevaplarının tamamını karşılamaktadır. Bu da metriğin kullanılabilir olduğu anlamına gelir.

Kod satır sayısı: Bu bilgiyi anlık olarak değil, her bir sürümün sonunda bilmekteyiz. Her bir sürüm alındığında, o sürüme ilişkin SVN'de bir etiket açılarak kaynak kodlar buraya yüklenir. Bu etiketlere istediğimiz zaman SVN'den

ulaşabilmekteyiz. Bu sayede bir kod satır sayısı sayma uygulaması aracılığı ile ilgili sürümün ne kadar kod satır sayısına sahip olduğunu görebilmekteyiz.

Çizelge 5.3. Kod Satır Sayısı Kullanılabilirlik Anketi

			Lütfen her bir niteliği, sorulara verilen cevaplar doğrultusunda aşağıda verilen dört ölçekle derecelendiriniz:			
Metrik Adı: Kod Satır Sayısı			F : Niteliğe ait göstergeler tamamen sağlanıyor (%86-100)			
Kavramsal Tanım: Her bir sürüme ait kodun satır sayısı cinsinden büyüklüğüdür.			L : Niteliğe ait göstergeler büyük oranda sağlanıyor (%51-85)			
Değerlendirme Tarihi : 01/03/2014			P : Niteliğe ait göstergeler kısmen sağlanıyor (%16-50)			
Değerlendiren: Anıl AYDIN			N : Niteliğe ait göstergeler yok ve sağlanamıyor (%0-15)			
Nitelikler			**Cevaplar**	**Derece**	Beklenen Cevaplar	
	Göstergeler					
Metrik Kimliği			**MKF-1**	**F**		
	S1	Metrik, hangi varlığı ölçüyor?	Kaynak kod			
	S2	Metrik varlığın hangi niteliğini ölçüyor?	Satır sayısı			
	S3	Metrik verisinin ölçeği nedir? (nominal, sıralı, aralıklı, oransal, mutlak)	Oransal		Oransal, Mutlak	
	S4	Metrik verisinin birimi nedir?	Satır			
	S5	Metrik verisinin türü nedir? (integer, real, etc.)	Sayı			
	S6	Metrik verisinin aralığı nedir?	4-6 haftada bir alınan her bir sürüm için			
Veri Varlığı			**MKF-2**	**F**		
	S7	Metrik verisi mevcut mu?	Evet		Mevcut > 20	
	S8	Tüm gözlemlerin miktarı nedir?	Bir araçla satır sayısı sayılıyor.			
	S9	Eksik veri noktalarının miktarı nedir?	Yok			
	S10	Veri noktaları belirli periyodlar halinde mi eksik? (Evet ise, gözlemlenen bu periyodların sayılarını belirtiniz)	Hayır			
	S11	Metrik verisi zaman sıralı mıdır? (Hayır ise, metrik verisinin nasıl sıralandığını belirtiniz)	Evet			
Veri Doğrulanabilirliği			**MKF-3**	**F**		
	S12	Metrik verisi sürecin hangi aşamasında kaydedilmiştir? (başlangıçta, ortada, sonda, daha sonra, vb.)	Her bir sürüm tamamlandığında			
	S13	Tüm metrik verileri sürecin aynı aşamasında mı kaydedildi? (başlangıçta, ortada, sonda, daha sonra, vb.)	Evet		Evet	
	S14	Metrik verisini kaydetmeden kim sorumludur?	Konfigürasyon Yöneticisi			
	S15	Tüm metrik verisi sorumlu kişi tarafından mı kaydedildi?	Evet		Evet	
	S16	Metrik verisi nasıl kaydedildi? (form ile, rapor ile, araç ile, vb.)	SVN'e kodlar her bir sürüm sonrası yükleniyor.			
	S17	Tüm metrik verileri aynı şekilde mi kaydedildi? (form ile, rapor ile, araç ile, vb.)	Evet		Evet	
	S18	Metrik verileri nerede saklanıyor? (bir dosyada, veritabanında, vb.)	SVN'de			
	S19	Tüm metrik verileri aynı yerde mi saklanıyor? (bir dosyada, veritabanında, vb.)	Evet		Evet	
Veri Bağımlılığı			**MKF-4**	**F**		
	S20	Metrik verisinin yaratılma sıklığı nedir? (asenkron, günlük, haftalık, aylık, vb.)	4-6 haftada bir			
	S21	Metrik verisinin kaydedilme sıklığı nedir? (asenkron, günlük, haftalık, aylık, vb.)	4-6 haftada bir			
	S22	Metrik verisinin saklanma sıklığı nedir? (asenkron, günlük, haftalık, aylık, vb.)	4-6 haftada bir			
	S23	Verilerin yaratılma, kaydedilme ve saklanma sıklıkları farklı mı?	Hayır		Hayır	
	S24	Metrik verisi eksiksiz olarak kaydediliyor mu?	Evet		Evet	
	S25	Metrik verisi belirli bir amaç için mi toplanıyor?	Evet		Evet	
	S26	Metrik verisinin toplanma amacı süreci yürüten kişiler tarafından biliniyor mu?	Evet		Evet	
	S27	Metrik verisi analiz edilip raporlanıyor mu?	Evet		Evet	
	S28	Metrik verisi analiz sonuçları süreci yürüten kişilerle paylaşılıyor mu?	Evet		Evet	
	S29	Metrik verisi analiz sonuçları süreci yönetimle paylaşılıyor mu?	Evet		Evet	
	S30	Metrik verisi analiz sonuçları karar verme amaçlı kullanılıyor mu?	Evet		Evet	
Veri Normalize Edilebilirliği						
	S31	Metrik verisi parametreler veya metriklerle normalize edilebilir mi? (Evet ise, hangileri belirtiniz)	Hayır			
Veri Entegre Edilebilirliği						
	S32	Metrik verisi proje düzeyinde entegre edilebilir durumda mıdır?	Evet			
	S33	Metrik verisi kurumsal düzeyde entegre edilebilir durumda mıdır?	Evet			

Yukarıda doldurulan anket sonuçlarını değerlendirecek olursak; doldurulan değerler beklenen anket cevaplarının tamamını karşılamaktadır. Bu da metriğin kullanılabilir olduğu anlamına gelir.

Hata yoğunluğu: Bu metrik yukarıda bahsi geçen hata sayısı metriğinin kod satır sayısı metriğine bölünmesiyle elde edilir(hata sayısı/kod satır sayısı).

Çizelge 5.4. Hata Yoğunluğu Kullanılabilirlik Anketi

			Lütfen her bir niteliği, sorulara verilen cevaplar doğrultusunda aşağıda verilen dört ölçekle derecelendiriniz:		
Metrik Adı: Hata Yoğunluğu			F : Niteliğe ait göstergeler tamamen sağlanıyor (%86-100)		
Kavramsal Tanım: Kod satır sayısı başına düşen hata sayısıdır.			L : Niteliğe ait göstergeler büyük oranda sağlanıyor (%51-85)		
Değerlendirme Tarihi: 01/03/2014			P : Niteliğe ait göstergeler kısmen sağlanıyor (%16-50)		
Değerlendiren: Anıl AYDIN			N : Niteliğe ait göstergeler yok ve sağlanamıyor (%0-15)		
Nitelikler			**Cevaplar**	**Derece**	Beklenen Cevaplar
	Göstergeler				
Metrik Kimliği			**MKF-1**	**F**	
	S1	Metriğin formülü nedir? (lütfen ilişkili taban metriklere referans veriniz)	Hata sayısı / Kod satır sayısı		
	S2	Metrik verisinin ölçeği nedir? (nominal, sıralı, aralıklı, oransal, mutlak)	Oransal		Oransal, Mutlak
	S3	Metrik verisinin birimi nedir?	Hata sayısı / Kod satır sayısı		
	S4	Metrik verisinin türü nedir? (integer, real, etc.)	Ondalıklı sayı		
	S5	Metrik verisinin aralığı nedir?	Proje başından itibaren her bir sürüm başına		
Veri Varlığı			**MKF-2**	**F**	
	S6	Metrik verisi mevcut mu?	Evet		Mevcut > 10
	S7	Tüm gözlemlerin miktarı nedir?	Proje başından itibaren her sürüm için hesaplanabilir		
	S8	Eksik veri noktalarının miktarı nedir?	Yok		
	S9	Veri noktaları belirli periyodlar halinde mi eksik? (Evet ise, gözlemlenen bu periyodların sayılarını belirtiniz)	Hayır		
	S10	Metrik verisi zaman sıralı mıdır? (Hayır ise, metrik verisinin nasıl sıralandığını belirtiniz)	Evet		
Veri Doğrulanabilirliği			**MKF-3**	**F**	
	S11	Metrik verisi nasıl hesaplanıyor? (araçla, elle, vb.)	Elle		
	S12	Tüm metrik verileri aynı şekilde mi hesaplanıyor? (araçla, elle, vb.)	Evet		Evet
	S13	Tüm metrik verileri metrik formülüne göre mi hesaplanıyor?	Evet		Evet
	S14	Metrik verileri nerede saklanıyor? (bir dosyada, veritabanında, vb.)	Kurumsal kalite ekibi tarafından saklanıyor		
	S15	Tüm metrik verileri aynı yerde mi saklanıyor? (bir dosyada, veritabanında, vb.)	Evet		Evet
Veri Bağımlılığı			**MKF-4**	**F**	
	S16	Metrik verisi eksiksiz olarak kaydediliyor mu?	Evet		Evet
	S17	Metrik verisi belirli bir amaç için mi toplanıyor?	Evet		Evet
	S18	Metrik verisinin toplanma amacı süreci yürüten kişiler tarafından biliniyor mu?	Evet		Evet
	S19	Metrik verisi analiz edilip raporlanıyor mu?	Evet		Evet
	S20	Metrik verisi analiz sonuçları süreci yürüten kişilerle paylaşılıyor mu?	Evet		Evet
	S21	Metrik verisi analiz sonuçları süreci yönetimle paylaşılıyor mu?	Evet		Evet
	S22	Metrik verisi analiz sonuçları karar verme amaçlı kullanılıyor mu?	Evet		Evet
Veri Normalize Edilebilirliği					
	S23	Metrik verisi parametreler veya metriklerle normalize edilebilir mi? (Evet ise, hangileri belirtiniz)	Hayır		
Veri Entegre Edilebilirliği					
	S24	Metrik verisi proje düzeyinde entegre edilebilir durumda mıdır?	Evet		
	S25	Metrik verisi kurumsal düzeyde entegre edilebilir durumda mıdır?	Evet		

Yukarıda doldurulan anket sonuçlarını değerlendirecek olursak; doldurulan değerler beklenen anket cevaplarının tamamını karşılamaktadır. Bu da metriğin kullanılabileceği anlamına gelir.

Çalışmada kullanılacak bu üç metriğin kullanılabilirliğinin değerlendirilebilmesi için doldurulmuş anketler göz önüne alındığında, sorulara verilen cevaplar, bu metriklerin kullanılabilir olduğunu göstermektedir.

5.2.1.3. Hata dağılımlarının gözlenmesi ve tahminleme modelinin seçilmesi

Gerçekleştirilen tahminleme çalışmaları boyunca kullanılacak modellerin seçilmesi işlemi, tahminlemeyi etkileyen en büyük etmenlerden biridir. Tahminleme hedefine uymayan ve tahminlenecek verinin doğasını yansıtmayan modellerin bu sürece olabildiğince dâhil edilmemesi gerekmektedir. Kurumların yazılım geliştirme maliyetini en aza indirgeme ve hızlı ürün elde edip müşteriye sunma çabaları nedeniyle ne yazık ki tahminleme sürecine yeteri kadar kaynak ayrılmamaktadır. Bu nedenle ilgili kurumların geleceğe yönelik hedeflerini ve kaynak planlarını etkilibir şekilde yapmaları için gerekli olan tahminleme sürecinin, kurumları cezp edici şekilde kolay, basit, hızlı ve anlaşılır olarak yürütülmesinin sağlanması gerekmektedir. Daha öncede bahsedildiği üzere her kurumun, süreçlerinin ve hedeflerinin farklı niteliklere sahip olması nedeniyle tahminleme

gerçekleştirecekleri proje veya bileşenler üzerinde kullanılacak modeli, en iyi şekilde kendi ihtiyaçları doğrultusunda belirlemeleri gerekmektedir. Bunun için de tahminleme gerçekleştirecek kurumun ilgili modeli seçme aşamasında bir karar mekanizmasının yer alması gerekir. Bu karar, genellikle proje yöneticisi veya kuruma ait kalite ekipleri tarafından ilgili veriler incelenerek verilmektedir.

Literatürde birçok tahminleme modeli yer almakta, bu çalışmaların geçerliliği bir şekilde bir veri kümesi üzerinde doğrulanarak ilgili çalışmalarda anlatılmaktadır. Ancak, bu kadar çok tahminleme modelinin varlığı, tahminleme sürecini gerçekleştirecek kişi ve kurumların aklının karışmasına ve karar verme aşamasının zorlaşmasına neden olmaktadır. Model seçimini yapacak ve kullanılacak modeli belirleyecek kişinin bu konuda yapılan çalışmaları incelemesi, anlaması ve uygulaması oldukça vakit gerektiren bir süreçtir.

Biz bu çalışmada, literatürde önerilmiş ve genellikle tahminleme amacıyla kullanılan istatistiksel modeller ile güvenilirlik modellerini inceledik. Model seçiminde dikkat ettiğimiz en önemli kriterler sırasıyla;

- Hata yoğunluğu bilgisinin zaman ekseni üzerinde gösterdiği dağılıma en yakın dağılımı gösteren bir model olması,
- Tahminleme hedeflerimiz doğrultusunda kolay ve anlaşılır bir şekilde tahminleme yapmayı sağlayan bir model olması,
- Modelin ele aldığı varsayımlarının ilgili tahminleme verilerine uygun varsayımları olan bir model olması,
- Modele ait kısıtların ilgili tahminleme sürecimizi aksatmayacak kısıtlar içeren bir model olması ve
- Kurumsal düzeyde bulunan kalite ekibimizin tahminleme sürecinde faydalandığı tahminleme modellerinin incelenmesidir.

Daha önce de bahsedildiği gibi üzerinde çalışılan hataların, bir zaman ekseni üzerinde gösterdiği dağılım, ilgili hataların projenin ilerleyen zamanlarında göstereceği dağılım hakkında bize bilgi vermektedir. Bu nedenle, hata dağılımlarının sahip olduğu örüntü, tahminleme modelimizin seçiminde etkin bir rol oynar.

Yaptığımız çalışmada, ilgili projeye ait hem modül seviyesinde hem de proje seviyesindeki hatalara ilişkin hata yoğunluklarının dağılımları gözlemlenmiş ve

analiz edilmiştir. Modül seviyesinde ve proje seviyesinde hata yoğunluklarının sürümler boyunca farklı dağılımlar izlediği gözlemlenmiştir. Bu nedenle proje seviyesi ve modül seviyesi açısından izlendiğinde hata yoğunluklarının tahminlenmesi sürecinde farklı modellere ihtiyaç duyulduğu fark edilmiştir.

İteratif yazılım geliştirmenin doğası gereği, her bir iterasyonda yazılıma eklenen yeni özelliklerden dolayı yeni hataların ortaya çıkması ve daha önceki iterasyonlardan kalan hata sayılarının azalması nedeniyle, proje düzeyindeki hata yoğunluğunun genellikle sabit bir seyir gösterdiği gözlemlenmiştir. Bunun sonucu olarak, proje düzeyindeki hataların iterasyonlar boyunca benzer seviyelerde kalması nedeniyle gösterdiği kümülatif hata yoğunluğu grafiği, belirli bir eğime sahip düz bir çizgi şeklinde yani lineer bir dağılım göstermiştir. Bu nedenle, hata yoğunluklarını en iyi şekilde ifade etmesi nedeniyle, proje seviyesinde yeni sürümler için gerçekleştirilen hata tahminleme işlemleri için Basit Lineer Regresyon Modeli'nin kullanılması tercih edilmiştir. Ancak, kümülatif hata yoğunluklarında son sürümlere doğru bir düşüş gözlemlenmesinden dolayı Rayleigh Kümülatif Dağılım Fonksiyonun'dan faydalanmak ve proje seviyesinde kümülatif hata yoğunluklarını Rayleigh dağılımı açısından da analiz etmek gerektiği düşünülmüştür. Bunun için proje seviyesinde 29 iterasyona ait hata yoğunluklarının analiz edilmesine ve bir tahminleme sürecinin yürütülmesine karar verilmiştir.

Modül seviyesinde kullanılacak olan tahminleme modelinin belirlenmesi ve bir tahminleme sürecinin oluşturulmasından önce, projede yer alan 14 modüle ait hata yoğunluklarının sürümler boyunca gösterdiği dağılım analiz edilmiştir. Bu 14 modülden 10 tanesinin Rayleigh Modeli'ne ait dağılım sergilediği gözlemlenmiştir. Kalan 4 modülün ise ağırlıklı olarak S-biçimli modele ait dağılıma benzer bir dağılım gösterdiği izlenmiştir. Bu 4 modülün daha tamamlanmamış olması ve diğer modüllere göre daha az veri noktasına sahip olması nedeniyle analiz sürecine katılmamasına karar verilmiştir. Yani başka bir deyişle, bu 4 modüle ait hatalar, daha az veri noktası(sürüm sayısı) ile S-biçimli bir dağılım göstermiştir. Modüllerin ele alındığı sürüm sayısı arttıkça gösterdikleri dağılım şekli, yeni modüller olmaları nedeniyle değişebileceği için bu modüller analiz sürecine katılmamıştır. Bunun aksine Rayleigh dağılımı gösteren 10 modül genellikle sonlamış ya da sonlanmaya yakın ve daha fazla sürüm boyunca geliştirilmiş ve ele alınmış yani daha fazla veri

noktasına sahip modüller olması nedeniyle analiz sürecinde daha anlamlı sonuçlar üreteceği için tercih edilmiştir. Modül bazındaki hata yoğunluklarının Rayleigh Modeli benzeri bir dağılım göstermesinin nedeni, zamanla ilgili modüle ait gereksinimlerin tamamlanması ve bu nedenle modüle ait ortaya çıkmış hataların çoğunun düzeltilmiş olması ve hata seyrinde bir düşüş gözlenmesidir. Rayleigh Modeli, hataların bir süre boyunca artıp tavan noktasına eriştikten sonra yavaş yavaş zamanla azaldığı bir eğri şeklinde bir dağılıma sahiptir. Bu nedenle proje seviyesindeki ve modül seviyesindeki hatalar farklı dağılım göstermiştir. Proje seviyesinde, sisteme her iterayonda farklı modüller veya farklı özellikler eklenmesi nedeniyle hata yoğunluğu belirli bir düzeyde sabitlenmiştir.

Teknik bilgiler kısmında yer alan güvenilirlik modelleri çoğunlukla literatürde bahsi geçen ve çeşitli hata verileriyle tahminlemeler yapılmış modellerdir. Ancak, bu güvenilirlik modellerinin en temel ön şartı tahminlenecek verinin özellikle test olmak üzere ortaya çıktığı geliştirme faz bilgisine ihtiyaç duymalarıdır [9]. Ancak, bizim çalışmamıza konu olan projede hataların hangi geliştirme fazına ait olduğu bilgisi yer almamaktadır. Bu nedenle daha çok test fazına ait hatalarla ilgilenen güvenilirlik modellerinin bu çalışmada tercih edilmemesinin temelinde bu yatmaktadır. Çizelge 5.5'de ilgili modellerin sahip oldukları varsayımlar ile kısıtları gösteren bir çizelge sunulmuştur.

Çizelge 5.5. Tahminleme Modellerinin Varsayım ve Kısıtları

Varsayım ve Kısıtlar	Schneidewind Modeli	Genelleştirilmiş Üssel Model	Musa/Okumoto Logaritmik Poisson Modeli	Littlewood/Verrall Modeli	Duane Modeli	Yamada, Ohba, Osaki S-Biçimli Güvenilirlik	Jelinski/Moranda Güvenilirlik Büyüme Modeli	Rayleigh Modeli	Lineer Regresyon Modeli
1) Bir zaman aralığında bulunan hata sayısı diğer bir zaman aralığında bulunan hata sayısından bağımsızdır.	x	x	x		x	x	x		
2) Sadece yeni gelen bir hata veya hata türü sayılır.	x					x			
3) Hata düzeltme oranı, yazılımda düzeltilen hata sayısı ile doğru orantılıdır.	x								
4) Yazılım her zaman beklenen gerçek çalışma ortamı ile aynı durumlarda çalışır.	x	x	x		x	x	x		
5) Ortalama bulunan hata sayısı, bir zaman aralığından başka bir zaman aralığına düşüş gösterir.	x	x							
6) Hata bulma oranı, yazılımda bulunan hata sayısı ile doğru orantılıdır.	x						x		
7) Hata bulma oranı üssel olarak bir düşüş gösterir.	x		x						
8) Tüm hatalar aynı oluşma ihtimaline sahiptirler.		x					x		
9) Her hata aynı önem derecesine sahiptir.		x					x		
10) Karşılaşılan hatalar yeni hatalara neden olmadan direk düzeltilir.		x				x	x		
11) Hata verileri toplanırken ve hata tahminleme işlemleri gerçekleştirilirken ilgili yazılım sistemi çalışmaya devam etmektedir.				x					
12) Her bir hata oluşma aralıkları arasında hata oranı sabit kalır.							x		
13) Dağılım eğrisi ne kadar yüksekse hata oranı da o kadar yüksektir.								x	
14) Geliştirme aşamasında ne kadar erken hata tespiti yapılır ve düzeltilirse sonraki aşamalarda da bir o kadar daha az hata ile karşılaşılır.		x						x	
15) Hatalar lineer dağılıma yakın bir dağılım sergiler.									x

Bir tahminleme süreci yürütecek kurum ve kişiler için;

- Hataların gösterdiği dağılımlar ve bu dağılıma ait örüntüler,
- İhtiyaç duyulan tahminleme verilerinin varlığı veya durumu,
- Projenin alan bilgisi ve geliştirme süreçleri,
- Tahminleme hedefleri,
- Tahminleme sonrası elde edilen sonuçların doğruluğu,
- Tahminleme modelinin performansı, kullanılabilirliği, basit olması ve öğrenilebilirliği

vb. birçok kriterin yanında seçilecek modelin;

- Amacı,
- Tahminleyebildiği veriler,
- Gerektirdiği parametreler ve yapısı,
- Varsayımları,
- Kısıtları,
- Veri gereksinimleri,
- Uygulandığı örnek çalışmalar,

incelenerek proje veya tahminlemenin gerçekleştirileceği bileşenin ihtiyaçları doğrultusunda ilgili modelin seçilmesi işlemi gerçekleştirilmelidir.

Rayleigh Modeli literatürde tahminleme amacıyla birçok çalışmaya konu olmuştur. Kullanımı ve parametre değerlerinin hesaplanması kolay bir modeldir. Yazılım geliştirme sürecini zamanın bir fonksiyonu olarak modelleyebildiği için hata yoğunluğu dağılımının da bu şekilde gösterilebilmesine olanak vermektedir. Modül seviyesinde hata yoğunluğunun; sıralı sürüm numaraları zaman birimi olarak temel alındığında gösterdiği dağılımın Rayleigh dağılımına benzediği görülmektedir. Hata yoğunluklarının da aynı dağılımı takip edeceği düşünülerek ileriye yönelik tahminlemeler yapmak mümkün olmaktadır.

Lineer Regresyon Modeli, çok basit bir matematiksel fonksiyona sahiptir. “y=ax+b” şeklinde bir fonksiyonla bize ‘x’ zaman ekseni boyunca ‘y’ ekseni ile ifade edilen hata yoğunluklarını tahminlememizde yardımcı olmaktadır. Bu çalışmada kullanılan projenin geliştirildiği kuruma ait kalite ekibinin de Lineer Regresyon Modeli’nden faydalanması bu modelin seçiminde etkili olmuştur. Rayleigh ve Lineer Modeller parametre hesaplamaları kolay olduğu için dağılıma ait tahminleme modellerinin yaratılması ve ilgili hesaplamaların yapılması işlemi de kolaylaşmaktadır. Bu çalışmada Rayleigh Modeli ile Lineer Regresyon Modeli’nin kullanımı uygun görülmüştür ve tahminleme sürecinde bu iki modelden faydalanılmıştır.

5.2.1.4. Araştırma sorularının belirlenmesi

Hem proje hem de modül seviyesindeki hata yoğunluklarının gösterdiği dağılımlar dikkate alınıp, bu dağılımlara ait en uygun modellerin seçilmesi işleminden sonra, proje ve modül bazlı hata yoğunlukları için tahminleme modellerinin kurulması ve bu modellerin ürettiği sonuçların gerçek sonuçlar açısından ele alınarak model performanslarının test edilmesi hedeflenmiştir. Bu nedenle, bu tez çalışması, Giriş bölümünde belirtilen ikinci araştırma sorusu özelleştirilerek tanımlanan bir takım araştırma hedefleri doğrultusunda gerçekleştirilmiştir ve bu doğrultuda aşağıda verilen araştırma soruları temel alınmıştır:

AS-1: İteratif bir yazılım projesine ait hata yoğunluklarının, Lineer Regresyon Modeli ile gerçekleştirilecek tahminleme işleminin performans düzeyi nedir?

AS-2: İterasyonlar boyunca proje düzeyindeki hata yoğunluklarının Rayleigh Modeli ile tahminlenmesi işleminin performans düzeyi nedir?

AS-3: İteratif olarak geliştirilen bir yazılım projesine ait modül seviyesindeki hata yoğunluklarının Lineer Regresyon Modeli ile gerçekleştirilen tahminleme işleminin performans düzeyi nedir?

AS-4: İterasyonlar boyunca modül düzeyindeki hata yoğunluklarının Rayleigh Modeli ile tahminlenmesi işleminin performans düzeyi nedir?

5.2.2. Modelin Oluşturulması

Bu başlık altında tahminleme amacıyla kullanılacak modellerin oluşturulma süreçleri detaylı olarak anlatılmıştır.

5.2.2.1. Tahminleme analiz prosedürlerinin çıkartılması

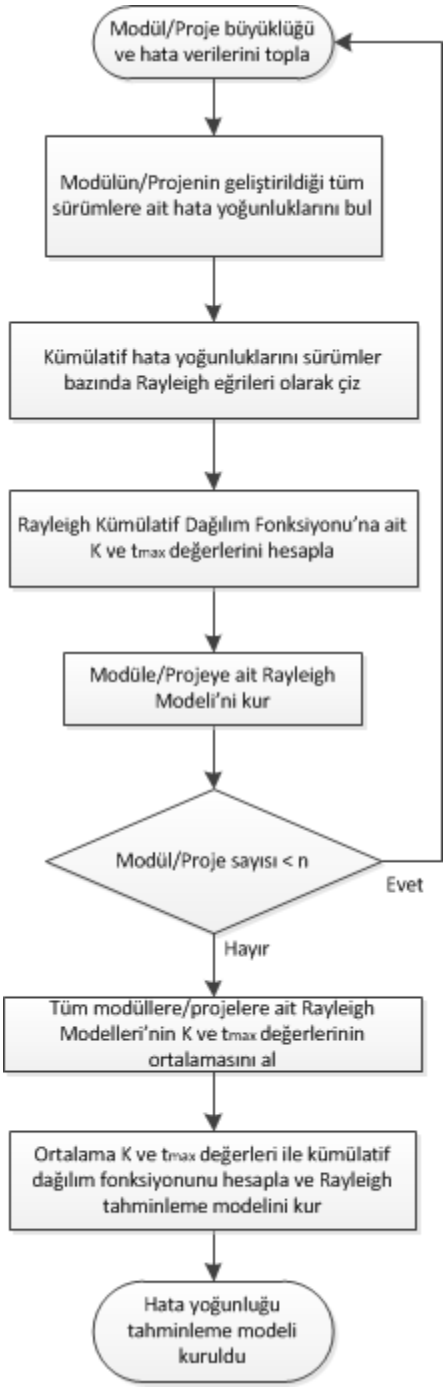
İlgili projeye ait hata yoğunluğunu tahminlemek için kullanılan modeller oluşturulurken, bir takım süreçlerden geçilmesi gerekmektedir. Bu başlık altında tahminleme modellerinin kurulması ile ilgili süreçleri daha detaylı açıklayabilmek için adım adım oluşturulmuş analiz prosedürlerinden bahsedilmiştir. Bu prosedür adımları, ilgili tahminleme modelinin oluşturulması aşamasında gerçekleştirilen faaliyetlerin genel bir özetini vermektedir. Bu çalışmada kullanılan Rayleigh ve Lineer modellere ait analiz prosedürleri aşağıdaki şekillerde adım adım gösterilmektedir.

Her iki prosedürde de iki adet parametrik değerden bahsedilmektedir. İki modelin analiz prosedürlerinin adımlarında yer alan ‘n’ parametresi, modellerin kurulması aşamasında modelleri eğitmek ve model katsayılarının belirlenmesi için kullanılan modül ya da proje sayısıdır. Bu çalışmada 3 adet modül ve 1 adet proje verileri için ilgili tahminleme modelleri kurulmuştur. Bu yüzden ilgili çalışmada ‘n’ değeri modül sayısı kastedildiğinde 3; proje sayısı kastedildiğinde 1 değerine karşılık gelmektedir. Yine her iki prosedürde yer alan ‘m’ parametresi, tüm sürümlerin sayısının yüzde olarak değerine denk gelmektedir. Bizim çalışmamız için ‘m’ parametre değeri 70’e denk gelmektedir. Yani, prosedüre bakıldığında modüllerin ve projelerin geliştirildiği tüm sürümlerin %70’i tahminleme modellerinin eğitilmesi ve modele ait katsayıların belirlenmesi için kullanılacaktır anlamına gelir. Tüm sürümlerin diğer %30’luk kısmına ait hata yoğunluğu verileri ise kurulan model ile

gerçekleştirilen tahminleme sonuçlarının karşılaştırılması ve modelin tahminleme performansının test edilmesi amacıyla kullanılmıştır.

Aşağıda hem Rayleigh hem de Lineer tahminleme modellerinin kurulması esnasında kullanılabilecek ikişer adet analiz prosedürleri verilmiştir. Bunlardan ilki(Şekil 5.3) hem modül hem de proje seviyesindeki hata yoğunluklarını, birden fazla modül veya proje için tahminlemek amacıyla kullanılabilecek olan analiz prosedürüdür. Bu analiz prosedüründe kullanılan birden fazla modül veya projenin tahminleme modellerine ait katsayıların ortalamaları alınarak yeni bir tahminleme modelinin kurulması hedeflenir. İkinci prosedür(Şekil 5.4) ise yine hem modül hem de proje seviyesindeki hata yoğunluklarını, sadece bir modül veya proje için tahminlemek amacıyla kullanılabilecek olan analiz prosedürüdür. Bu prosedür de ise ilgili modül veya projeye ait hata yoğunluğu verilerinin belirli bir yüzdesi kullanılarak eğitilen bir tahminleme modeli kurulur ve ilgili modül veya projenin kalan kısımlarına ait verilerin tahminlenmesi hedeflenir.

Rayleigh modeli prosedür adımlarına bakacak olursak;



Şekil 5.3. Birden Fazla Modül veya Proje için Kullanılan Rayleigh Modeli Analiz Prosedürü

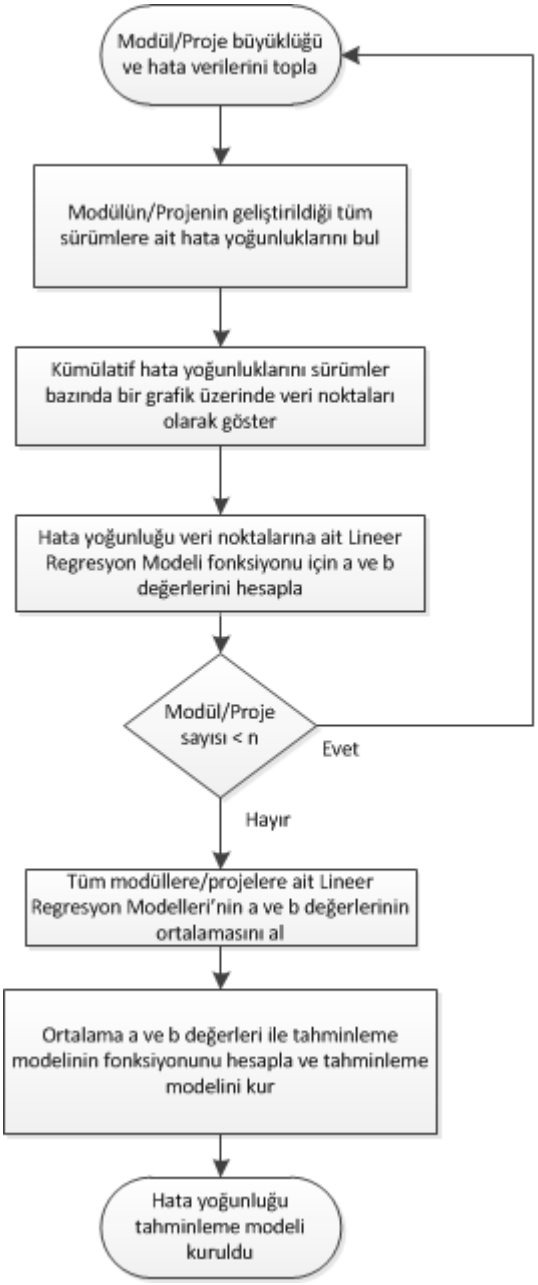
Şekil 5.4. Verilerin Belirli Bir Yüzdesi için Kullanılan Rayleigh Modeli Analiz Prosedürü

Bu prosedürleri adım adım açıklayacak olursak;

- Modül veya projeye ait her sürüm için kod satır sayısı cinsinden büyüklük bilgisi ile yine aynı modül veya projeye ait hata sayısı verileri toplanır.
- Her bir sürüm için ilgili modül veya projeye ait kaydedilen hata sayısı yine ilgili sürümlerdeki modül veya proje büyüklüğüne(kod satır sayısı) bölünerek her bir sürümdeki hata yoğunluğunu bilgileri bulunur.
- İlgili hata yoğunluklarına ait kümülatif hata yoğunluğu – sürüm numarası grafiği çizilir.
- Eğer birden fazla modül veya proje ile çalışılıyorsa her bir modül veya proje için Rayleigh Kümülatif Dağılım Fonksiyonu'nda yer alan 'K' ve 'tmax' değerleri hesaplanır. Sadece bir modül veya proje kullanılıyorsa ilgili modül veya projeye ait %m'lik kısma ait hata yoğunluğu verileri için 'K' ve 'tmax' değerleri hesaplanır.
- Birden çok modül veya proje için kurulan tüm Rayleigh modellerine ait kümülatif hata yoğunluğu fonksiyonları için hesaplanan 'K' ve 'tmax' değerlerinin ortalaması alınır ve hesaplanan ortalama 'K' ve 'tmax' değerleri ile tahminleme için kullanılacak Rayleigh Modeli'nin kümülatif dağılım fonksiyonu belirlenir. Tahminleme modeli kurulur.
- Tahminleme için kurulan bu Rayleigh modeline ait kümülatif dağılım fonksiyonu ile tahminlemeler yapılır.

Bu çalışmada, kurulan Rayleigh tahminleme modeli için modül seviyesinde birden çok modül kullanıldığı için ilk analiz prosedürünün; proje seviyesinde ise sadece bir projeye ait verilerden faydalanıldığı için ikinci analiz prosedürünün kullanılması tercih edilmiştir.

Lineer model prosedür adımlarına bakacak olursak;



Şekil 5.5. Birden fazla modül veya Proje için Kullanılan Lineer Model Analiz Prosedürü

Şekil 5.6. Verilerin Belirli Bir Yüzdesi için Kullanılan Lineer Model Analiz Prosedürü

Bu prosedürleri adım adım açıklayacak olursak;

- Modül veya projeye ait her sürüm için kod satır sayısı cinsinden büyüklük bilgisi ile yine aynı modül veya projeye ait hata sayısı verileri toplanır.

- Her bir sürüm için ilgili modül veya projeye ait kaydedilen hata sayısı yine ilgili sürümlerdeki modül veya proje büyüklüğüne(kod satır sayısı) bölünerek her bir sürümdeki hata yoğunluğunu bilgileri bulunur.
- İlgili hata yoğunluklarına ait kümülatif hata yoğunluğu – sürüm numarası grafiği çizilir.
- Eğer birden fazla modül veya proje ile çalışılıyorsa her bir modül veya proje için Lineer Regresyon Modeli Fonksiyonu'nda yer alan 'a' ve 'b' değerleri hesaplanır. Sadece bir modül veya proje kullanılıyorsa ilgili birime ait %m'lik kısma ait hata yoğunluğu verileri için 'a' ve 'b' değerleri hesaplanır.
- Birden çok modül veya proje için kurulan tüm Lineer modellere ait kümülatif hata yoğunluğu fonksiyonları için hesaplanan 'a' ve 'b' değerlerinin ortalaması alınır ve hesaplanan ortalama 'a' ve 'b' değerleri ile tahminleme için kullanılacak Lineer Model'in kümülatif dağılım fonksiyonu belirlenir. Tahminleme modeli kurulur.
- Kurulan bu Lineer Model fonksiyonu ile tahminlemeler yapılır.

Bu çalışmada, kuralan Lineer tahminleme modeli için modül seviyesinde birden çok modül kullanıldığı için ilk analiz prosedürünün; proje seviyesinde ise sadece bir projeye ait verilerden faydalanıldığı için ikinci analiz prosedürünün kullanması tercih edilmiştir.

5.2.2.2. Tahminleme modelinin kurulması hazırlığı

Tahminleme amacıyla kuracağımız modellerin eğitilmesi ve oluşturulması amacıyla verilerin bir kısmının ayrıştırılması gerekmektedir. Ayrıştırılan bu veriler yardımıyla ilgili tahminleme modellerinin fonksiyonları belirlenmiştir. Bu amaçla, proje seviyesindeki tahminleme modellerini eğitmek amacıyla tüm projeye ait 29 adet sürümün %70'ine ait yani 20 sürüme ait hata yoğunluğu verileri, modellerin eğitilmesi amacıyla kullanılmıştır. Kalan %30'luk sürümlere ait yani 9 sürüme ait hata yoğunluğu verileri tahminleme sonuçlarının gerçeğe yakınlığının değerlendirilmesi amacıyla kullanılmıştır. Modül seviyesinde tahminleme yapabilmek amacıyla 3 adet modüle ait hata yoğunluğu verileri, tahminleme modellerinin eğitilmesi amacıyla; 1 adet modüle ait hata yoğunluğu verileri, kurulan tahminleme modellerinin sonuçlarının değerlendirilmesi amacıyla kullanılmıştır.

5.2.2.3. Tahminleme modelinin kurulması

a) Modül Seviyesi Tahminleme Modellerinin Kurulması

Bu başlık altında analiz prosedürlerinde Rayleigh Modeli için verilen adımlar takip edilerek Rayleigh Kümülatif Dağılım Fonksiyonu kullanılmış ve ilgili projeye ait M1, M2 ve M3 olarak adlandırılan 3 modülün tahminleme modelleri kurulmuştur. Bu üç modüle ait hata yoğunlukları gözlemlendiğinde Rayleigh Modeli'nin sahip olduğu dağılıma benzer bir örüntüde dağılım gösterdikleri görülür. Modüllere ait Rayleigh modelleri kurulurken modelin kümülatif dağılım fonksiyonundaki toplam hata yoğunluğu değerine denk gelen 'K' değeri 2.7 formülündeki %40 prensibinden [37] faydalanılarak hesaplanmaktadır. Yani, başka bir deyişle ilgili çalışmada anlatıldığı gibi hata yoğunluklarının en yüksek seviyeye ulaştığı tmax anında tüm hataların %40'ı ortaya çıkmıştır ve hata yoğunluğu bu anda en yüksek seviyededir. Çizelge 5.6'da, toplam hata yoğunlukları ve hatanın en üst seviyeye ulaştığı tmax anından faydalanılarak oluşturulmuş M1, M2 ve M3 modüllerinin 2.5 eşitliğiyle gösterilen Rayleigh Kümülatif Dağılım fonksiyonlarına ait eşitlikler verilmiştir.

Çizelge 5.6. M1, M2 ve M3 Modüllerine ait Rayleigh KDF Eşitlikleri

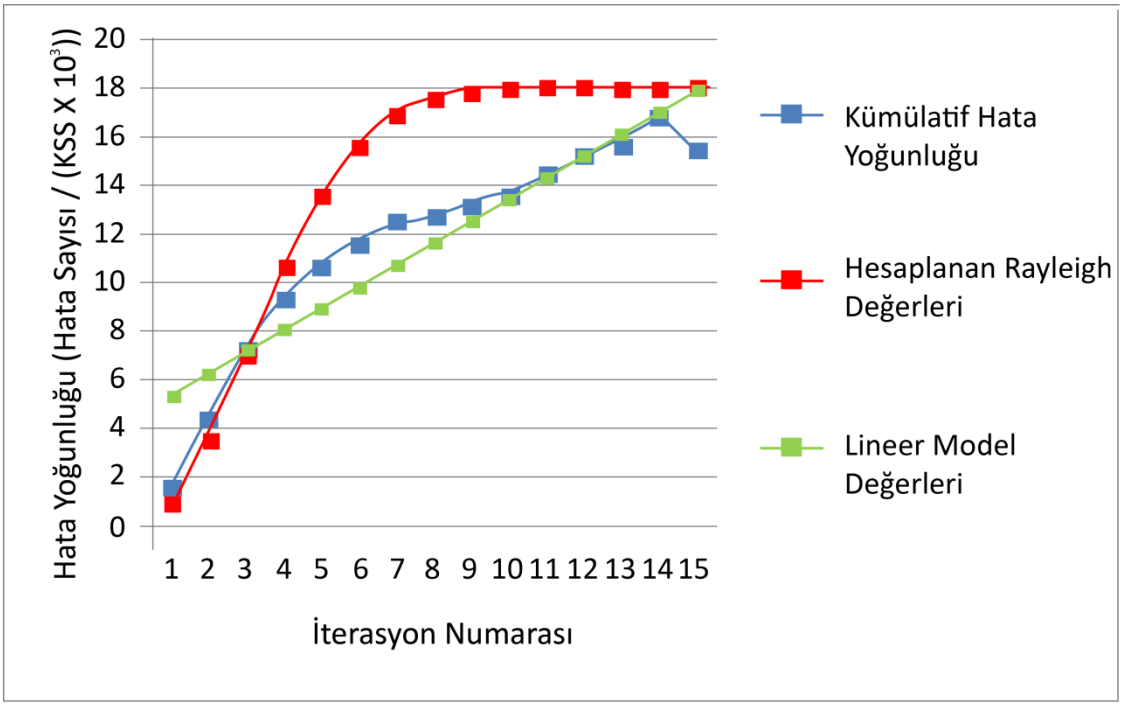
Modül Adı	Rayleigh KDF Modeli Eşitliği
M_1	$F(t) = 18{,}10\ x\ (1 - e^{-(t/18)^2})$
M_2	$F(t) = 13{,}53\ x\ (1 - e^{-(t/98)^2})$
M_3	$F(t) = 9{,}25\ x\ (1 - e^{-(t/72)^2})$

M1, M2 ve M3 modüllerine ait hata yoğunluklarını tahminlemek için Rayleigh modelleri kurulduktan sonra, aynı hata yoğunluğu verileri ile ilgili modüllere ait Lineer Regresyon Modelleri de kurulmuştur. M1, M2 ve M3 modülleri için kurulan Rayleigh Modelleri ve Lineer Modeller, 15 sürüm ya da diğer adıyla 15 iterasyon için kurulmuş ve test edilmiştir. Lineer Modeller için 2.5. Lineer Regresyon Modeli başlığı altında verilen 2.9, 2.10 ve 2.11 eşitliklerinden faydalanılarak bu üç modül için 15 sürüm boyunca ortaya çıkan hata yoğunluklarıyla oluşturulmuştur. Çizelge 5.7'de M1, M2 ve M3 modülleri için kurulan Lineer Modellere ait eşitlikler verilmiştir.

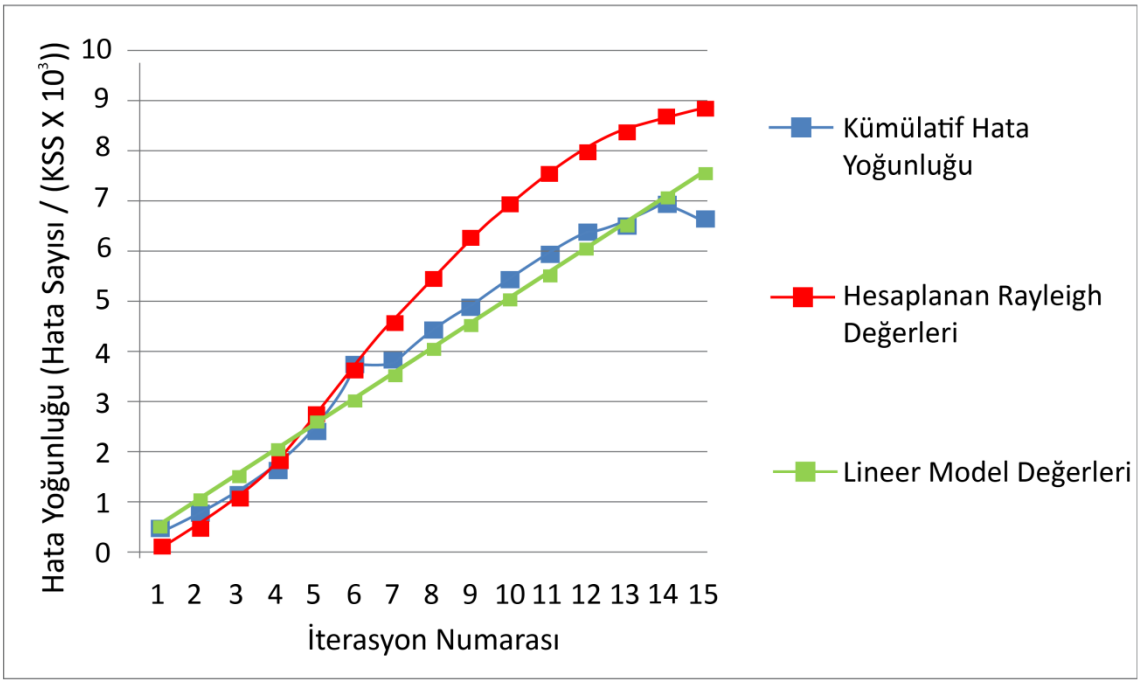
Çizelge 5.7. M1, M2 ve M3 Modüllerine ait Lineer Model Eşitlikleri

Modül Adı	Lineer Model Eşitliği
M_1	$y = 4{,}437 + 0{,}899x$
M_2	$y = 0{,}057 + 0{,}5x$
M_3	$y = 1{,}593 + 0{,}474x$

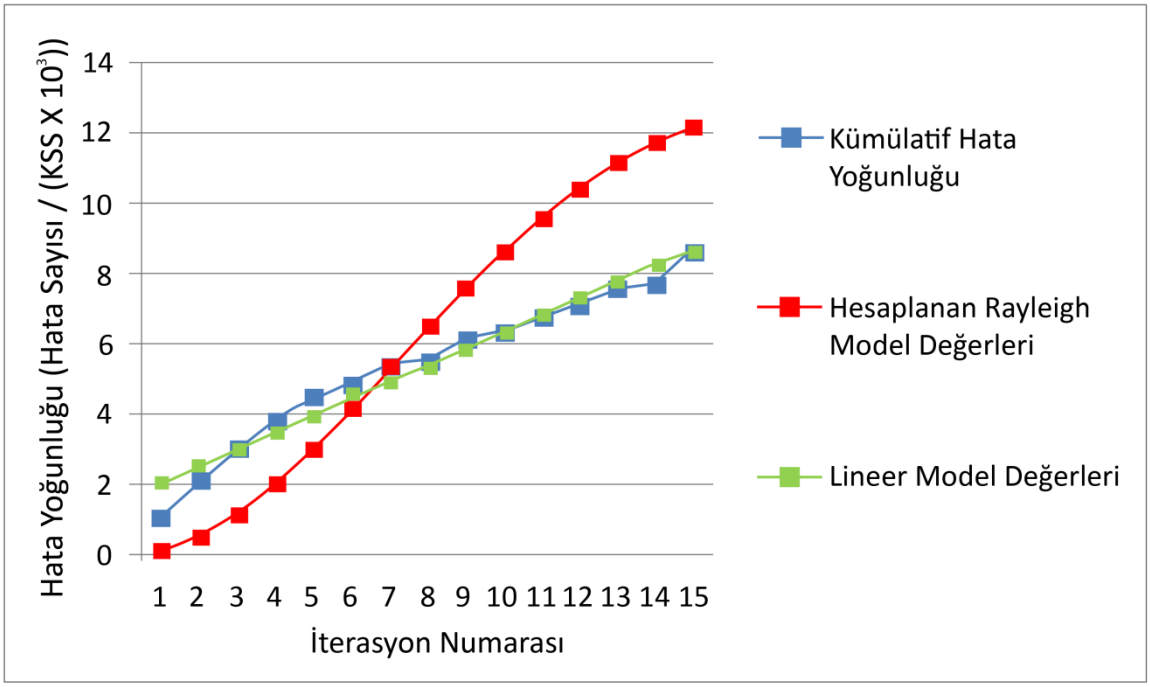
Şekil 5.7, 5.8 ve 5.9'da, sırasıyla M1, M2 ve M3 modüllerine ait 1-15 sürümleri boyunca gerçek kümülatif hata yoğunluğu dağılımları ile kurulan modeller aracılığıyla hesaplanan değerler doğrultusunda oluşan kümülatif hata yoğunluğu dağılımları verilmiştir.



Şekil 5.7. M1 Modülüne ait Hata Yoğunluğu Dağılımları



Şekil 5.8. M2 Modülüne ait Hata Yoğunluğu Dağılımları



Şekil 5.9. M3 Modülüne ait Hata Yoğunluğu Dağılımları

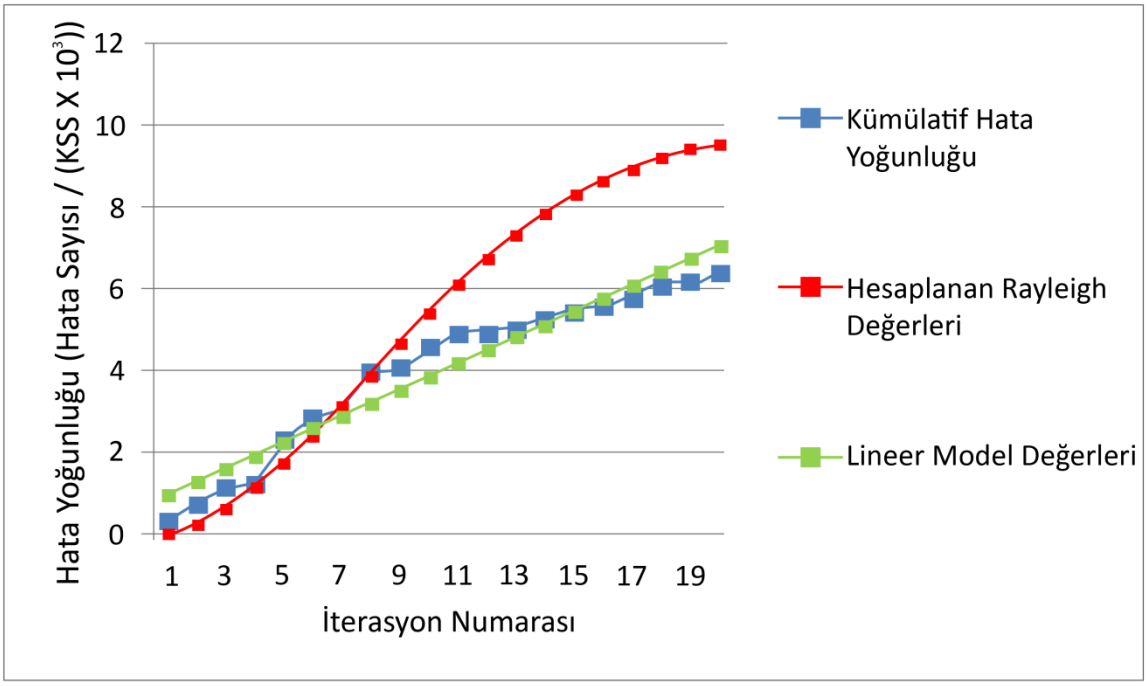
b) Proje Seviyesi Tahminleme Modellerinin Kurulması

Modül seviyesi modelleri kurulduktan sonra, proje seviyesindeki hata yoğunluğu verileri kullanılarak Lineer ve Rayleigh modelleri kurulmuştur. Buradaki modeller, 29 sürüm ya da iterasyon için kurulup test edilmiştir. Bunun için tüm bu 29 sürümün %70'i (20 sürüm) modeli eğitmek amaçlı; %30'u (9 sürüm) kurulan modellerin performansının test edilmesi amaçlı olacak şekilde bölünmüştür. Modül seviyesi Rayleigh modellerinin kurulması esnasında 2.5 eşitliğindeki Rayleigh fonksiyonunda yer alan 'K' toplam hata yoğunluğu değeri, proje seviyesi Rayleigh modeli için de aynı şekilde 2.7 eşitliğinde gösterilen %40 prensibi [37] ile hesaplanarak bulunmuştur. Yine aynı şekilde 2.5. Lineer Regresyon Modelleri başlığı altında verilen 2.9, 2.10 ve 2.11 eşitliklerinden faydalanılarak 20 sürüm için Lineer Model eşitliği hesaplanmıştır. Çizelge 5.8'de proje seviyesinde kurulmuş Rayleigh ve Lineer Modellerine ait eşitlikler verilmiştir.

Çizelge 5.8. Proje Seviyesi Model Eşitlikleri

Model Adı	**Model Fonksiyon Eşitliği**
Rayleigh KDF Modeli	$F(t) = 10{,}04\, x\, (1 - e^{-(t/128)^2})$
Lineer Model	$y = 0{,}682 + 0{,}319x$

Şekil 5.10'da ise proje seviyesindeki 1-20 sürümleri boyunca her bir sürüme ait gerçek kümülatif hata yoğunluğu dağılımları ile kurulan modeller aracılığıyla hesaplanan değerler doğrultusunda hesaplanan kümülatif hata yoğunluğu dağılımları bulunmaktadır.



Şekil 5.10. Proje Seviyesi Hata Yoğunluğu Dağılımları

5.2.3. Tahminleme

Bu aşamada, kurulan modeller doğrultusunda tahminleme ve tahminleme sonuçlarının değerlendirilmesi adımları gerçekleştirilmiştir.

5.2.3.1. Tahminlemenin gerçekleştirilmesi

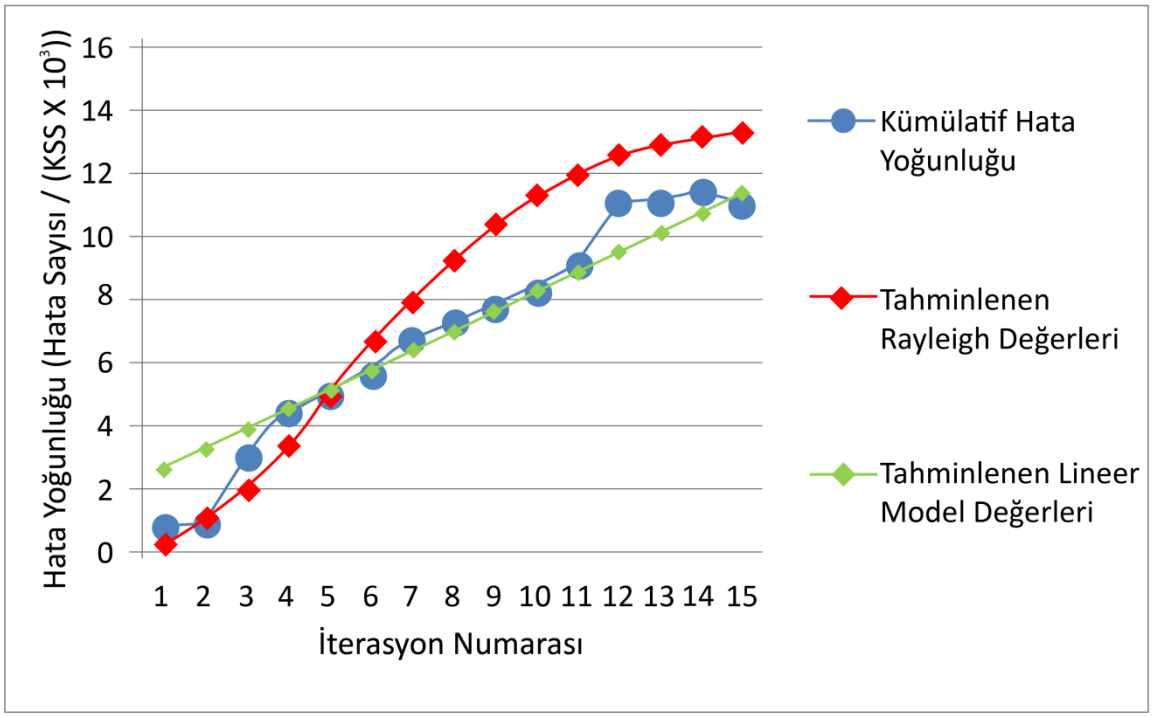
Modül seviyesinde, Rayleigh Modeli başlığı altında verilen Rayleigh Kümülatif Dağılım Fonksiyonu eşitliği ile M1, M2 ve M3 modülleri için modeller kurulmuştur. Bu üç modüle ait kurulan modellerin fonksiyonlarındaki toplam hata yoğunluğuna denk gelen 'K' değerleri ile hata yoğunluğunun maksimum seviyeye ulaştığı tmax zamanlarının değerlerinin ortalamaları alınarak yeni bir Rayleigh Kümülatif Dağılım Fonksiyonu eşitliği oluşturulmuştur. Bu ortalama değerlerle oluşturulan eşitlik modül seviyesinde kurulan Rayleigh tahminleme modeli eşitliğine denk gelmektedir. Bu eşitlik ile kurulan Rayleigh tahminleme modeli diğer üç modül haricinde Mp olarak adlandırılan başka bir modüle ait tahminlemenin yapılması için kullanılmıştır. Mp modülü için yapılan tahminleme işlemleri sonucunda, ortaya çıkan hata yoğunluğu değerleri ile Mp modülüne ait gerçek hata yoğunluğu değerleri karşılaştırılmıştır.

Rayleigh Modeli haricinde, tahminleme yapabilmek için bir de Lineer Regresyon Modeli kurulmuştur. Bunun için M1, M2 ve M3 modüllerine ait oluşturulan lineer fonksiyon eşitliklerinden faydalanılmıştır. Bu üç modül için Lineer Regresyon Modeli başlığı altında bahsedilen eşitliklerde yer alan 'a' ve 'b' değerleri, daha önceki başlıkta hesaplanmıştı. Mp modülü için oluşturulan Lineer tahminleme modeli eşitliğinde bu üç modüle ait eşitliklerde yer alan 'a' ve 'b' değerlerinin ortalaması alınarak yeni bir eşitlik yaratılmıştır. Çizelge 5.9'da Mp modülüne ait gerçekleştirilen tahminleme işlemleri için oluşturulmuş Rayleigh Modeli ve Lineer Regresyon Modeli eşitlikleri verilmiştir.

Çizelge 5.9. Mp Modülüne ait Model Eşitlikleri

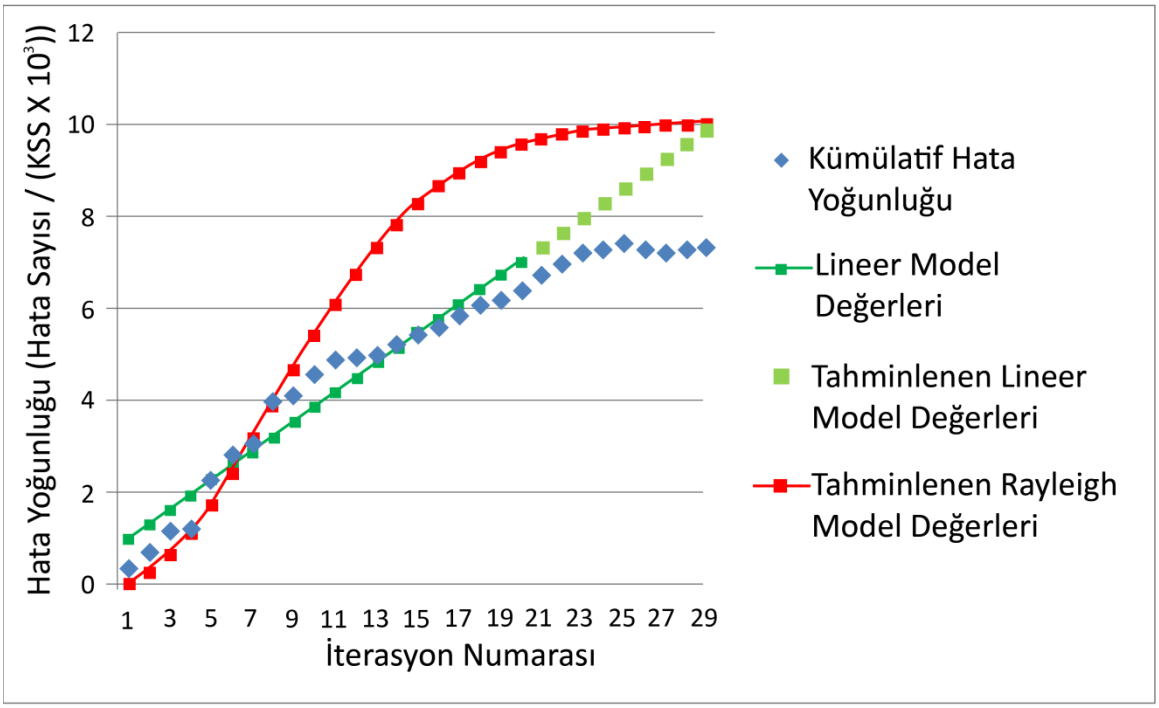
Model Adı	**Model Fonksiyon Eşitliği**
Rayleigh KDF Modeli	$F(t) = 13{,}63\ x\ (1 - e^{-(t/56)^2})$
Lineer Model	$y = 2{,}027 + 0{,}624x$

İlgili Mp modülüne ait tahminleme modellerine ilişkin eşitlikler hesaplandıktan sonra, her bir sürüm yani her bir iterasyon için kümülatif hata yoğunluğu değerleri, bu eşitlikler ile hesaplanmıştır. Şekil 5.11, Mp modülüne ait gerçek hata yoğunlukları ile tahminlenmiş hata yoğunluklarının 1-15 iterasyonlarına ait dağılımlarını göstermektedir.



Şekil 5.11. Modül Seviyesi Tahminleme Modelleri Dağılımları

Bu adımdan sonra, proje seviyesine ait tahminleme modelleri kurulmuştur. İlk olarak 29 adet sürümün %70'i yani 20 sürüm ile Lineer Model ve Rayleigh Model eğitilmiş ve kurulan modellerle kalan 9 sürüm için tahminleme yapılmıştır. Proje seviyesinde yapılan tahminlemeler proje seviyesi modelinin kurulması başlığı altında verilen Çizelge 5.8'deki eşitlikler kullanılarak yapılmıştır. Şekil 5.12'de, 9 adet sürüm için tahminlenen hata yoğunluğu veri noktaları da eklenerek, tüm hata yoğunluklarının dağılımları gösterilmektedir. Şekilden, gerçek hata yoğunlukları ile tahminlenmiş hata yoğunlukları izlenebilir.



Şekil 5.12. Proje Seviyesi Tahminleme Modelleri Dağılımları

Tüm bu dağılımlara ait sonuçlara bakıldığında kurulan tahminleme modellerinin hesapladığı tahminleme değerleri, gerçek hata yoğunluğu değerlerinden biraz daha fazla hesaplanmaktadır. Buradan, geliştirme yapan ekibin etkili ve az hataya sebep olacak şekilde çalıştığı veya gerçekleştirilen test faaliyetleri sonucu yeterince hata bulunamadığı bu nedenle daha az hatanın bulunduğu çıkarımı yapılabilir.

5.2.3.2. Tahminleme sonuçlarının ve model performanlarının analiz edilmesi

Modül seviyesi ve proje seviyesi hata yoğunluğu tahminleme modelleri kurulduktan sonra, kurulan modellerin ne kadar iyi çalışıp gerçeğe yakın sonuçlar ürettiğini değerlendirebilmek için ilgili modellerin tahminleme performanslarının ölçülmesi gerekmektedir. Literatürde tahminleme modellerinin performanlarını değerlendirmede kullanılan birçok metrik bulunmaktadır. Kurulan modellerin uyum derecelerinin (“goodness of fit”) ölçülmesi için şu metrikler tercih edilmiştir; Karekök Ortalama Hata(“Root Mean Square Error” – “RMSE”), Ortalama Mutlak Hata(“Mean Absolute Error” – “MAE”), Bağıl Hata(“Magnitude Relative Error” – “MRE”), Ortalama Bağıl Hata(“Mean Magnitude of Relative Error” – “MMRE”)’dır.

RMSE, ilgili modelin gerçeğe yakın sonuçlar ürettiğini değerlendirmek amacıyla ortalama hata değerlerinin büyüklüğünü ölçer [69]. Bunun için gerçek değerler ile tahmin edilen değerler arasındaki fark ölçülür ve bu farkların karelerinin ortalaması alınır. Daha sonra bu ortalama değerin karekökü alınarak RMSE değeri hesaplanmış olur. RMSE'nin eşitliği şöyledir;

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - f_i)^2}{n}} \tag{5.1}$$

MAE de, RMSE gibi tahminleme modelinin gerçeğe yakınlığını, ortalama hata değerlerinin büyüklüğünü ölçerek bulur [69]. Ancak, MAE için hataların yönü önemlidir. MAE ölçümünün sonucu, tahminlenen değerlerin gerçek değerlere ne kadar yakın olduğunu gösterir. RMSE değerleri ile MAE değerleri arasında çok az bir fark olması beklenir. Bu fark ne kadar küçükse hataların varyasyonlarının da o derece küçük olduğu anlaşılır. MAE'ye ait eşitlik şöyledir;

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - f_i| \tag{5.2}$$

MRE, tahminlenen değerlerle gerçek değerlerin arasındaki bağıl hata büyüklüğünü yüzde olarak hesaplamayı sağlar. Bu metriğin eşitliği;

$$MRE = \frac{|y_i - f_i|}{y_i} \tag{5.3}$$

MMRE, tahminlenen değerlerin gerçek değerlerden ne kadar farklılık gösterdiğini ölçerek ilgili modelin performansını değerlendirmeyi sağlar. MMRE değerinin ≤ 0.25 olması bize modelin iyi bir performansa sahip olduğunu ve modelin veriyi iyi tahminleyebildiğini gösterir [70]. MMRE eşitliği şöyledir;

$$MMRE = \frac{\sum_{i=1}^{n}\frac{|y_i - f_i|}{y_i}}{n} \tag{5.4}$$

MRE ile MMRE arasındaki ilişki;

$$MMRE = \frac{\sum_{i=1}^{n} MREi}{n} \tag{5.5}$$

şeklindedir. MRE ile her bir veri noktasının mutlak bağıl hatasını inceleyebiliriz. PRED(q) ise belirli bir MRE değeri(q) ve altında yer alan tahminleme noktalarının yüzdesini veren bir metriktir ve veri noktaları açısından tahminlemenin ne kadar gerçeğe yakın olduğunu görmemizi sağlar. PRED(q) eşitliği şöyledir;

$$PRED(q) = \frac{i}{n} \tag{5.6}$$

Burada i değeri $MRE_i \leq$ q olan veri noktalarının sayısını; n ise toplam veri noktalarının sayısını vermektedir. Bir referansa göre [71], MMRE ≤ 0,25 ve PRED(0,25) ≥ 0,75 olan sonuçlar için tahminleme modelinin başarılı olduğu söylenmiştir.

Çizelge 5.10'da, modül seviyesinde kurulan Rayleigh ve Lineer tahminleme modellerinin RMSE, MAE ve MMRE değerleri bulunmaktadır.

Çizelge 5.10. Modül Seviyesi Tahminleme Modelleri Performans Değerleri

	Rayleigh KDF Modeli	**Lineer Model**
RMSE	1,777	0,969
MAE	1,515	0,664
MMRE	0,239	0,375

Yukarıdaki çizelgede görüldüğü üzere, modül seviyesinde Lineer Modelle yapılan tahminleme sonuçlarına ait RMSE ve MAE değerlerinin Rayleigh Modeli'ne oranla daha düşük olduğu görülmektedir. Özellikle Rayleigh Modeli olmak üzere her iki modelde RMSE ve MAE değerleri arasındaki farkın küçük olduğu yani sıfıra yakınsadığı görülebilir. Bu farkın az olması da hataların varyasyonlarının küçük olduğu anlamına gelmektedir. Ancak, RMSE ve MAE değerlerine göre Lineer Model daha iyi bir performansa sahip gibi görünse de, Lineer modele ait MMRE değerinin, kabul edilebilir değer olan 0.25'den büyük olduğu görülmektedir. Rayleigh Modeli'ne ait MMRE değeri ise 0.25 değerinden küçük olduğu için daha iyi performans gösterdiği ve bu nedenle modül seviyesi hata yoğunluğunu tahminlemede daha uygun bir model olduğu düşünülebilir.

Çizelge 5.11'de ise proje seviyesinde gerçekleştirilen tahminleme için Rayleigh ve Lineer Modellerine ait RMSE, MAE ve MMRE değerleri yer almaktadır.

Çizelge 5.11. Proje Seviyesi Tahminleme Modelleri Performans Değerleri

	Rayleigh KDF Modeli	Lineer Model
RMSE	2,707	1,592
MAE	2,413	1,434
MMRE	0,375	0,197

Yukarıdaki çizelge incelenecek olursa, proje seviyesindeki hata yoğunluğu tahminleme sonuçları açısından Lineer Model'e ait tüm RMSE, MAE ve MMRE değerlerinin Rayleigh Modeli'ne göre daha iyi olduğu gözlemlenmektedir. Yine, RMSE ve MAE değerleri arasındaki farkın özellikle Lineer Model'de olmak üzere her iki model için de düşük olduğu yani sıfıra yakınsadığı, bu nedenle de hata varyasyonlarının küçük olduğu söylenebilir. Ancak, proje seviyesindeki tahminleme sonuçlarına ait MMRE değerlerine bakıldığında Lineer Model'e ait değerin Rayleigh Modeli'ne göre daha iyi olduğu, Lineer Model'in MMRE değerinin kabul edilebilir değer olan 0,25 değerinin altında; Rayleigh Modeli'ne ait değerinse 0,25 üstünde olduğu görülür. Bu nedenle proje seviyesinde gerçekleştirilecek olan tahminleme sürecinde Lineer Model'in daha gerçeğe yakın sonuçlar ürettiği söylenebilir.

İlgili projeyi meydana getiren modüller sürümler boyunca geliştirilmektedir ve bu modüllere ilişkin gereksinimler gerçekleştirilip tamamlandıkça, ilgili modüle ait ortaya çıkan hataların sayısında bir azalış meydana gelmektedir. Bu da modüllere ait kümülatif hata yoğunluğu dağılımının iterasyonlar bazında gösterdiği örüntünün Rayleigh kümülatif dağılım eğrisine benzemesine neden olmaktadır. Modül seviyesinde üç modül ile kurduğumuz Rayleigh Modeli ile gerçekleştirilen tahminlemede, üç modülden daha fazla modülden faydalanarak modelin performansının artırılması sağlanabilir.

Proje seviyesinde ise Lineer Model'in tahminleme sonuçlarının Rayleigh Modeli'nin tahminleme sonuçlarına göre daha kabul edilebilir olmasının nedeni hata yoğunluklarının Lineer Model'e yakınsamasıdır. Çünkü, her bir iterasyonda projeye eklenen yeni modüllerin, yeni özelliklerin veya yeni gereksinimlerin neden olduğu hata yoğunluk artışı, önceden geliştirilmiş ve hataları düzeltildiği için hata yoğunluğu azalan bileşenlerin hata yoğunluklarına eklenerek ortalama hata yoğunluğu aynı oranda kalmaktadır. Bu da, kümülatif hata yoğunluğunun sürümler bazındaki dağılımının bir doğru şeklinde lineer bir dağılım göstermesine sebep

olmaktadır. Bu yüzden proje seviyesindeki hata yoğunluklarının tahminlenmesi sürecinde Lineer Model'in kullanılması daha uygundur.

Çizelge 5.12'de modül ve proje seviyesinde Rayleigh ve Lineer Modeller için hesaplanmış PRED(0,25) değerleri bulunmaktadır.

Çizelge 5.12. Modül ve Proje Seviyesinde Tahminleme Modeli PRED(0,25) Değerleri

	Modül Seviyesi PRED(0,25)	**Proje Seviyesi PRED(0,25)**
Rayleigh Model	%60	% 0
Lineer Model	%80	% 66,6

PRED(0,25), değerlerini inceleyecek olursak, her bir veri noktası açısından bakıldığında tahminleme sonuçlarına göre bağıl hata değerleri 0,25 ve altında olan veri noktaları Lineer Model'de daha çoktur. Yani, veri noktaları bazında Lineer Modelin Rayleigh Modele göre gerçeğe daha yakın tahminlemeler yaptığı söylenebilir. Modül seviyesinde, Rayleigh Modeli veri noktalarının %60'ını bağıl hata değeri 0,25 ve altında olarak tahminleyebilirken; Lineer Model'de bu %80'e çıkmıştır. Proje seviyesinde ise Rayleigh tahminleme sonuçlarının hepsi 0,25 ve üzeri bağıl hataya sahip olması nedeniyle PRED(0,25) değeri %0 iken, Lineer Model tüm veri noktalarının %66,6'sı için 0,25 ve altında MRE değerine sahip tahminlemeler gerçekleştirmiştir. Daha önce de bahsedildiği gibi MMRE değeri 0,25 ve altında olup PRED(0,25) değeri 0,75 ve üzerinde olan bir tahminleme sonucu bulunmamaktadır. Ancak, modül seviyesinde her iki tahminleme modeli için, proje seviyesinde de sadece Lineer Model için tüm veri noktalarının büyük çoğunluğunun 0,25 ve altı MRE değerine sahip olduğu söylenebilir.

6. ÇALIŞMANIN GEÇERLİLİĞİNİ ETKİLEYEBİLECEK HUSUSLAR

Gerçekleştirilen bir durum çalışmasına ait yapısal geçerliliğin(“Construct Validity”), iç ve dış geçerlilik tehditlerinin (“Threats to Internal and External Validity”) ve sonuç geçerliliğinin(“Conclusion Validity”) ya da güvenilirliğinin(“Reliability”) değerlendirilmesi ve analiz edilmesi önemli bir süreçtir [72]. Yapısal geçerlilik, gerçekleştirilen çalışmaya ait ölçümlerin doğru belirlenmesi ile ilgilidir. Yeterli metriklerin toplanması ve bu metriklerin toplanması sürecinin öznellikten uzak bir şekilde gerçekleştirilmesi gerektiğiyle ilgilenir. Gerçekleştirilen çalışmada, GQM yöntemi [40] ve ISO 15939 [41] standardı kullanılarak hata verilerinin toplanması, sürecin belirli standart ve yöntemler doğrultusunda gerçekleştirilmesini sağlamıştır. Kabul edilmiş bu yöntem ve standartlara dayandırılarak gerçekleştirilen ölçme, veri toplama ve değerlendirme süreçleri ile hedeflenen metriklerin doğru belirlenmesi ve kullanılması sağlanmıştır.

İç geçerlilik, durum çalışmaları için bir kalite test kriteridir [72]. Sapma içerebilecek durumları test etmek amacıyla da kullanılır. Genellikle sebep ve sonuç ilişkisine dayalı araştırma çalışmaları yürütülmektedir. Bağımlı ve bağımsız değişkenler arasında bir ilişki olduğunu savunur. İç geçerlilik bu çalışmalardaki; bağımsız değişkenlerde meydana gelen değişikliklerin bağımlı değişkenlerde gözle görülebilir derecede ne kadar etki ile sonuçlandığını, sonuçların ispatlarının güçlü ya da zayıf olmasını kontrol eder. Eğer bir çalışma yüksek derecede bir iç geçerliliğe sahipse biz de nedensellik için o derece güçlü kanıtlara sahip olmuş oluruz.

Bu çalışmanın iç geçerliliğinin tehditleri şöyle sıralanabilir:

- Sadece bir projeye ait hata yoğunluğu verileri kullanılmıştır, bu nedenle gerçekleştirilen çalışmanın, projelerin niteliklerinden etkilenip etkilenmediğini bilemeyiz. Bu yüzden, birden çok proje üzerinde aynı çalışmaları gerçekleştirerek bu durumu analiz etmeliyiz. Ayrıca, modül seviyesindeki hata yoğunlukları için gerçekleştirilen tahminleme sürecinde üç adet modül kullanılmıştır. Fakat, kullanılan bu üç modül haricindeki modüllere ait niteliklerin önerilen model üzerindeki etkilerini bilemiyoruz. Bu nedenle, bu etkinin araştırılabilmesi için daha fazla modülün analiz edilmesi

ve olası diğer tahminleme modellerinin de kullanılması bu tehdidi ortadan kaldırmada faydalı olacaktır.

- Bu çalışmada, proje seviyesinde hata yoğunluğu verilerinin %70'i tahminleme modelinin kurulması ve eğitilmesi aşamasında; %30'luk kısım ise tahminleme modelinin performansının ölçülmesi amacıyla kullanılmıştır. Hata yoğunluklarının farklı yüzdelerle kullanılıp modelin kurulması ve modelin performansının ölçülmesi sağlanarak en iyi sonucu verecek yüzde oranlarının belirlenmesi gerçekleştirilebilir. Böylece, modeli eğitmek için hangi seviyedeki yüzdelerin daha iyi olduğu çıkartılabilir.
- Çalışmada, kullanılan metriklerin kullanılabilirliğinin analiz edilebilmesi için MKA kullanılmıştır. MKA'nın bize sağladığı anketler yardımı ile gerçekleştirilen kullanılabilirlik analiz süreci, yaptığımız çalışmanın öznelliğinin azalmasını sağlamıştır. Ayrıca, bu yöntem ile tahminleme modeli kurulmadan önce kullanılacak verilerin varlığının kontrolü de sağlanmış olmaktadır. Ancak, ilgili metriklerin kullanılabilirliğinin değerlendirilmesi için tahminleme amacıyla kullanılacak verilerin ölçülmesi sürecinde gereken deneyim ve uzmanlık bilgisi, duruma bir derece öznellik katmaktadır. Projeyi geliştiren ekip kişisel uzmanlıklarına göre değerlendirilmemiştir.
- Bu çalışmada elde edilen tahminleme sonuçları, MMRE, RMSE ve MAE gibi ölçümler kullanılarak tahminleme modellerinin performanslarının ölçülmesi sağlanmıştır. Çalışmada tahminleme amacıyla kullanılan Rayleigh Modeli ile Lineer Model sonuçları karşılaştırılarak hangi modelin hangi durumda daha performanslı çalıştığının karşılaştırılması yapılmıştır. Ancak, bu karşılaştırma proje bağlamının aynı kalması nedeniyle anlamlıdır. Bu da iç tehdidi azaltan bir etmendir. Farklı projelere ait sonuçların karşılaştırılması proje bağlamları değişeceği için anlamsızlaşacaktır.
- Proje seviyesinde Lineer Modelin daha etkili çıkmasının sebebi projenin tamamlanmamış olmasıdır. Proje tamamlandıktan sonra gerçekleştirilecek tahminlemede hata yoğunluğu eğrisi bir azalış göstereceği için Rayleigh Modeli'ne yakınsayacaktır. Bu da Rayleigh tahminleme sonuçlarının gerçeğe daha yakın sonuçlar üretmesini sağlayacaktır.

Bir diğer kalite test kriteri ise gerçekleştirilen durum çalışmalarının genelleştirilebilirliğini test eden dış geçerliliktir [72]. Çalışmada elde edilen dış geçerlilik sonuçları şöyle sıralanabilir;

- Yapılan çalışmada, oluşturulan modellerin ne kadar doğru sonuçlar ürettiğini değerlendirmek ve model performanslarını ölçmek amacıyla daha önce de belirtildiği gibi MMRE, RMSE ve MAE kabul edilmiş ölçümleri kullanılmıştır. Gerçekleştirilen deney sonuçları çoğu çalışmada elde edilen sonuçların performansının ve doğruluğunun değerlendirilmesi aşamasında kullanılan bu ölçümler, kullanımlarının kolay olması ve sık kullanılması nedeniyle elde edilen değerlerin değerlendirilebilmesi açısından çalışmaya büyük kolaylıklar sağlamaktadır. Bu ölçümler, herhangi bir tahminleme çalışmasında elde edilen tahminlenmiş sonuçlarla gerçek değerlerin karşılaştırması yapılarak tahminleme sürecinin değerlendirilmesinde kullanılabilir. Ancak, bu çalışmada elde edilen model performans sonuçlarını genellemek mümkün değildir.
- Model seçimi aşamasında, hata yoğunluklarının bir zaman ekseni üzerinde gösterdiği dağılım ile alternatif modellerin varsayım ve kısıtları dikkate alınarak en uygun modelin seçimi yapılmıştır. Model seçiminin bu yöntemle gerçekleştirilmesi tüm değerlendirme kriterlerini kapsamayabilir. Bu kriterler haricinde model seçimini etkileyebilecek başka durumlar da olabileceği için model seçimi aşamasının kapsamının artırılması, bunun için örnek olarak bir kural tabanı ya da kriter listesinin geliştirilmesi ve bu kural veya kriter sonuçlarına göre uygun model seçiminin yapılmasını sağlayacak bir yöntemin belirlenerek sürecin genelleştirilmesi sağlanabilir.
- Önerilen tahminleme süreç modeli adımları, genel olarak süreci modellemesi nedeniyle, bu modeli kullanacak diğer kurumlar için tüm süreç boyunca izlenecek adımlar hakkında yönlendirici bir rol üstlenmektedir. Önerilen modele ait alt adımlar kurumlar tarafından kendilerine uyarlanarak gerçekleştirilebilir bir niteliğe sahiptir.
- Gerçekleştirilen çalışma, CMMI 3 seviyesinden CMMI 4 olgunluk seviyesine geçiş yapmış belli niteliklere sahip bir kurum tarafından geliştirilmiş bir proje üzerinde geliştirilmiştir. İlgili çalışma, bu olgunluk seviyesine ulaşmış ve

belirli süreçleri gerçekleştiren kurumlar için kullanılabilir bir örnek oluşturmaktadır.

Gerçekleştirilen çalışma için, kullanılan tüm metot ve yöntemlerin benzer hedefleri taşıyan çalışmalara uygulanabilir olduğu hakkında emin olamayız.

Sonuç geçerliliği veya güvenilirliği, önerilen çalışmaların ve analizlerin aynı sonuçlarla tekrarlanabilirliğinin kontrol edilmesini sağlar [72]. Bu kapsamda, gerçekleştirdiğimiz çalışmaların tekrar edilebilirliğini sağlamak amacıyla tahminleme süreci ve bu süreçte kullanılan tahminleme modellerinin kurulması süreci modellenerek, sürecin adım adım prosedürler halinde gerçekleştirilebilir olması sağlanmıştır. Ancak, bu prosedürlerin kurumsal seviyede kullanılabilirliğinin sağlanması ve iyileştirilmesi için daha fazla çalışma ile desteklenmesi gerekmektedir.

7. ÇIKARILAN DERSLER

Bu tez çalışması esnasında, bir takım zorluklarla karşılaşılmış ve bu problemleri çözerken edinilen deneyimler ve çıkarılan dersler, hata tahminleme ile ilgili çalışma yapacak araştırmacılara yardımcı olabileceği düşünülerek paylaşılmıştır.

Çalışmaya başlandıktan sonra ilgili projenin geliştiriliş şekli, doğası ile hata bulma ve düzeltme faaliyetlerinin nasıl gerçekleştirildiğini anlamak için çok zaman harcanmıştır. İlk olarak, birçok tahminleme ve güvenilirlik modellerinde ihtiyaç duyulan; kaydedilen hataların analiz, tasarım, kodlama, test veya bakım geliştirme fazlarından hangisinde ortaya çıktığı bilgisine göre hataları gruplanmış ve dağılımları gözlemlenmiştir. Ancak, hatalarda bu bilgilerin kaydedilmediği gözlemlenmiştir. Hataların hangi faza ait olduğu bilgisinin ilgili hataların içeriklerinin incelenerek çıkarılabileceğini tespit edilmiş; ancak hata sayısının çokluğu ve böyle bir işlemi bir araç yardımı olmadan gerçekleştirmenin zorluğu nedeniyle bu çıkarımdan vazgeçilmiştir. Hatalar, geliştirme faz bilgisine göre incelenemediği için daha önceki geliştirme yaşam döngülerine ait kalitenin sonraki yaşam döngüsü kalitesine olan etkisine ulaşmak mümkün olmamıştır. Bu nedenle iteratif olarak geliştirilen bir projede ortaya çıkan hataların kaydedilmesi esnasında, şelale modelindeki gibi ilgili hatanın hangi faza ait olduğunun kaydedilmesi gerektiği öğrenilmiştir.

Daha sonra, projeye ait hataların sürüm bazlı, bileşen bazlı, modül bazlı ve proje bazlı dağılımları çıkartılarak, grafikler üzerinde gözlemlenmiştir. İlgili hataların, tespit edilme haftalarına ya da 5'er, 10'ar ya da 20'şer günlük eşit aralıklara göre dağılımları çıkartılmıştır. Bileşen bazlı ve sürüm bazlı hata dağılımları gözlemlendiğinde, ilgili zaman birimlerine düşen hata verisi miktarının çok az olduğu izlenmiştir. Hatta bazı veri noktalarına hiç hata düşmediği, tüm bu eksik veri noktaları sebebiyle de ilgili hataların göstereceği dağılımın izlenebilirliğinin kaybolduğu gözlemlenmiştir. Üzerinde çalışılan projeye ait bileşenler modülleri; modüller de projeyi meydana getiren öğelerdir. Bu nedenle veri noktalarındaki veri miktarının artışını sağlamak amacıyla modül ve proje bazlı hataların kullanılması gerektiğini uygun görülmüştür.

Hata dağılımları gözlemlendikten sonra, ilgili hataların düzeltilmesi ve sistemden uzaklaştırılması için harcanan efor bilgisine ulaşılarak, ileriye yönelik hata düzeltme efor tahmininin yapılması hedeflenmiştir. Ancak, hata düzeltme efor bilgisine ait veriler, metrik kullanılabilirliği açısından incelendiğinde çoğu geliştiricinin bu bilgiyi kaydetmediği fark edilmiştir. Bu veriyi çok düzgün giren ekipler incelendiğinde bile hata düzeltme efor bilgilerinin, veri tabanı hataları için veya müşteri ortamında denenip düzeltilen hataların bir kısmı için eksik olduğu gözlemlenmiştir. İlgili ekip, kod geliştirme aracına kurduğu bir eklenti ile düzeltme faaliyetlerine ait efor bilgilerini kaydetmekteydi. Ancak, farklı makineler üzerinden müşteri ortamına bağlanıp düzeltilen hatalar için efor bilgilerinin girilmediği fark edildi. Bu nedenle hata düzeltme efor bilgisine ait tahminleme gerçekleştirilemedi.

MKA, tahminleme süreci için hangi metriğin kullanılabilir olduğunun anlaşılmasına yardımcı olmuştur. MKA'nın sunduğu anketler yardımıyla hata sayısının ve ürün büyüklüğü verisinin uygun olduğuna ve hata yoğunluğu (hata sayısı/ürünün kod satır sayısı cinsinden büyüklüğü) metriğinin bir tahminleme modeli kurma süreci için kullanılabilir olduğuna karar verilmiştir.

Ancak bu aşamada, sürüm kodlarına ulaşmada ve kod satır sayısını hesaplamada bir takım zorluklarla karşılaşılmıştır. Özellikle projeyi oluşturan bileşen ve modüllere ait kod satır sayısını eski sürümler için hesaplamak oldukça zahmetli bir işti. Çünkü eski sürümlerde, kodlar bileşen ve modül bazlı olarak gruplu değildi. İlgili kodlar, gerçekleştirilen iyileştirme çalışmaları sonrasında bu şekilde gruplandı. Eski sürümlere ait modül büyüklüklerinin hesaplanabilmesi için, ilgili modüle ait farklı paketlerdeki ilişkili kodların ayıklanması ve bulunması gerekti. Bu da tüm projede ilgili modüle ait kod dosyalarının belirlenmesi gibi bir efor gerektirdi. Tüm bu nedenlerden ötürü de kodlama faaliyetleri esnasında ilgili kodların düzenli bir biçimde paketlenmesi gerektiği öğrenildi. Böylece geliştirme yapacak veya ilgili kodları herhangi bir süreçte kullanacak kişilere kolaylık sağlanmış olacaktır. Diğer yandan da tüm proje büyüklüğü bilgisini bulmak oldukça kolay olmuştur. Çünkü bu seviyede tüm kodlara ait kod satır sayısının hiçbir ayrım yapılmadan ölçülmesi mümkündü.

Son olarak, tahminleme modelinin seçilmesi esnasında dikkat edilmesi gereken en önemli nokta, projenin hata dağılımına ve dağılımın gösterdiği davranışa ve örüntüye en uygun modelin seçilmesi işlemidir. Bunun için ilgili hataların sürümler boyunca gösterdikleri dağılımlar grafikler üzerinde gözlemlenmiş ve modül seviyesindeki dağılıma en uygun olarak Rayleigh modelinin; proje seviyesindeki dağılıma ise hata yoğunluklarının bir doğru olarak ilerlemesi nedeniyle Basit Lineer Regresyon Modeli'nin uygun olduğu tespit edilmiştir. Burada ilgili modellerin belirlenmesinde, hataların istatistiksel olarak gösterdikleri dağılım dikkate alınmıştır. Bu dağılıma sahip uygun istatistiksel modellerin kullanılması hataların ileride göstereceği dağılımın tahminlenmesinde de süreci kolaylaştıran etmenlerden biri oldu. Bunun haricinde, hata verilerinde kaydedildikleri geliştirme fazının bulunmaması ve Rayleigh ile Lineer Modellerin bu bilgiye ihtiyaç duymaması nedeniyle bu modeller tercih edilmiştir. Bu bilgiler doğrultusunda genel olarak şu söylenebilir; model seçimi esnasında literatürde bulunan tahminleme ve güvenilirlik modellerinin incelenerek ve ilgili modellerin varsayımları ve kısıtları göz önünde bulundurularak kullanılacak tahminleme model seçimi yapılmalıdır.

Yukarıda elde edilen deneyimler ve çıkarılan dersler detaylı olarak anlatılmıştır. Bunlar maddeler halinde toparlanacak olursa;

- Tahminlemede kullanılacak verilerin doğasını anlamak için gerekli çalışmalar yapılmalı ve yeterli süre ayrılmalıdır.
- Literatürde yer alan modellerin, hata tahminleme sürecinde daha etkin bir şekilde kullanılabilmesi için ilgili hatalara ait tespit edildiği geliştirme faz bilgisinin de kaydedilmesi iyi olacaktır. Böylece hangi geliştirme fazında ne gibi eksikliklerin olduğu bilgisi elde edilebilir, hatanın hangi aşamadan kaynaklandığının belirlenmesi sağlanabilir.
- Tahminlemede kullanılacak verilerin dağılımlarının, tahminleme sürecini kolaylaştıracak şekilde belirlenmesi gerekmektedir. Bu nedenle, verilerin uygun bir şekilde gruplandırılarak(haftalık, aylık, sürüm bazlı ya da bileşen, modül, proje bazlı vb.) dağılımlarının belirlenmesi iyi olacaktır.
- Hata düzeltme efor tahmini ve bunun sonucu proje takviminin tahminlenmesi ve planlanması yapılmak isteniyorsa, proje yönetimi ya da kurumsal düzeyde bu bilgilerin proje başlangıçlarında nasıl tutulması

gerektiği, ileride gerçekleştirilecek tahminleme hedefleri doğrultusunda belirlenmelidir.

- Tahminleme esnasında herhangi bir yetersiz veya eksik veri durumu ile karşılaşmamak için kullanılacak verilerin kullanılabilirlik ve uygunluk açısından değerlendirilmesi ve uygunsa kullanılması, olası zaman kayıplarının önüne geçecektir.
- Geliştirilen kodların projenin en başından itibaren mantıksal olarak bileşen, modül, vb. şekilde gruplu olarak paketlenmesi kodun yönetimi açısından kolaylık sağlayacaktır. Bu nedenle, kod-paket yönetiminin ve bu konuda gerekli iyileştirmelerin devamlı olarak yürütülmesi iyi olacaktır.
- Tahminlemede kullanılacak en uygun modelin seçilmesi sürecinde; tahminleme hedefleri, tahminlenecek verinin doğası, mevcut tahminleme modellerinin olanakları, kısıtları ve varsayımlarının dikkate alınması doğru tahminleme sonuçları elde etme açısından oldukça önemlidir.

8. SONUÇ VE ÖNERİLER

Bu tez çalışması kapsamında, hataların ortaya çıktığı geliştirme fazı bilgisinin kaydedilmemiş olduğu, iteratif olarak geliştirilmiş bir kamu projesine ait hata yoğunluğunun tahminlenmesi süreci üzerinde gerçekleştirilen çalışma anlatılmıştır. Bu amaçla, ilgili projenin modül ve proje düzeyindeki hata yoğunlukları analiz edilerek gelecek sürümlere ait hata yoğunluklarının tahmin edilmesi işlemleri gerçekleştirilmiştir. Modül seviyesindeki hata dağılımları incelenerek hataların daha küçük kod parçaları halinde; proje seviyesindeki dağılımlar incelenerek ise hataların daha genel olarak analiz edilmesi sağlanmıştır. Bunun için bir güvenilirlik modeli olan Rayleigh Modeli ile istatistiksel bir model olan Basit Lineer Regresyon Modeli kullanılmıştır. Her iki model de basit istatistiksel ve matematiksel bilgi gerektiren ve kendine özgü dağılıma sahip olan modellerdir. Modül seviyesindeki hata yoğunluklarının tahminlenmesi için üç ayrı modüle ait Rayleigh ve Lineer Model fonksiyonları çıkartıldıktan sonra bu fonksiyona ait katsayıların ortalamaları ile bir tahminleme modeli yaratılmış ve bu tahminleme modeli ile bir diğer modülün hata yoğunluklarının tahminlenmesi sağlanmıştır. Daha sonra, bu modüle ait gerçek hata yoğunluğu değerleri ile tahminlenen değerlerin karşılaştırması gerçekleştirilmiştir. Proje seviyesinde ise tüm sürümlerin %70'ine ait hata yoğunlukları kullanılarak Rayleigh ve Lineer Model tahminleme fonksiyonları yaratılmış ve kalan %30'luk kısmın hata yoğunlukları tahminlenerek gerçek değerlerle karşılaştırılması sağlanmıştır.

Tahminleme modeli kurmak amacıyla kullandığımız modüllerin her biri, birbirinden farklı karmaşıklığa sahip modüllerdir. Modül seviyesinde gerçekleştirilen tahminlemede daha fazla modül kullanılarak tüm modüllere ait ortalama değere yaklaşılacak ve diğer modülleri tahminleme sürecinde daha performanslı sonuçlar elde edilebilecektir. Proje seviyesinde, iteratif geliştirmenin doğası gereği hataların ortaya çıkma oranları genellikle sabit çıkmaktadır. Bunun en büyük nedeni, projeye yeni nitelikler eklemeyi sağlayan yeni modüllerin geliştirilmesi nedeniyle hata sayısının artması ve eski modüllere ait hataların da aynı oranda bir azalış göstermesidir. Hataların ortaya çıkma oranlarının sabit kalması, ilgili kümülatif hata ve hata yoğunluklarının Lineer Model'in gösterdiği lineer doğru şeklinde bir dağılıma sahip olmasına sebep olmaktadır. Bu nedenle, proje seviyesindeki hata yoğunlukları Lineer Model'in sahip olduğu dağılıma, Rayleigh Modeli'nin

dağılımına göre daha çok benzerlik göstermektedir. Modül seviyesinde ise ilgili modüle ait gereksinimlerin zamanla tamamlanması ve yeni özelliklerin gelmemesi nedeniyle ortaya çıkan hataların sayısında yaşanan azalma, kümülatif hata yoğunlukları dağılımının zamanla azalış gösteren bir eğriye sahip olmasına, bunun da Rayleigh Modeli'ne ait dağılıma benzemesine sebep olmaktadır.

Çalışma kapsamında, tahminleme doğrultusunda kullanılacak verilerin analiz edilmesi aşamasında literatürde kabul görmüş Hedef-Soru-Metrik, ISO/IEC 15939 gibi yöntem ve standartlardan faydalanılmıştır. Bu nedenle, verilerin değerlendirilmesi süreci kurumlara örnek teşkil etmektedir. Tahminlemede kullanılacak modellerin seçimi aşamasında literatürde yer alan güvenilirlik modelleri incelenmiştir ve hata yoğunluklarının gösterdikleri dağılımlara öncelik verilerek benzer dağılıma sahip Lineer Regresyon ve Rayleigh Modelleri'nin seçilmesine karar verilmiştir. Bu modellerin tercih edilmesindeki en önemli faktör hata yoğunluklarının bu modellerle benzer dağılıma sahip olmalarıdır. Ayrıca, ilgili kuruma ait kalite ekibinin de tahminleme sürecinde Lineer Model'den faydalanması bu modelin tercih edilmesinde etkili olmuştur. Sağladıkları bu kolaylıklar sayesinde, tahminleme sürecinde yapılacak hesaplamalar ve değerlendirmelerin en kolay ve en basit şekilde yapılabilmesi ve uygulanabilir olması hedeflenmiştir. Ayrıca, hataların kaydedildiği faz bilgisini gerektirmemeleri ve hata yoğunluklarının bir zaman ekseninde dağılımlarını değerlendirmeyi kolaylaştırmaları bu modellerin artıları olmuştur. Böylece, giriş bölümünde verilen ve ilk araştırma sorusu olan, hata kaydedilme faz bilgisinin bulunmadığı iteratif projeler için tahminleme sürecinde hangi modellerin daha uygun olduğu sorusuna cevap verilmiştir.

Proje seviyesindeki tahminleme sonuçları değerlendirilerek bu düzeyde Lineer Model'in Rayleigh Modeli'ne göre daha performanslı çalıştığı ve daha doğru sonuçlar ürettiği gözlemlenmiştir. Daha önceden bahsedilen ve MMRE değerinin 0,25'in altında olmasının kabul edilebilir olduğunu söyleyen bir çalışma [70] dikkate alınarak elde edilen sonuçlar değerlendirildiğinde; Lineer Model'in bu değerin altında bir MMRE değerine sahip olması nedeniyle, proje seviyesindeki tahminleme sürecinde Rayleigh Modeli'ne göre daha doğru sonuçlar ürettiği görülmüştür. Modül seviyesindeki sonuçlara baktığımızda ise RMSE ve MAE değerlerine göre Lineer Model daha performanslı gibi gözükse de MMRE değerinin Lineer Model için 0,25 değerinin üzerinde olması ve Rayleigh Modeli için

bu değerin kabul edilebilir değer aralığında bulunması nedeniyle, Rayleigh Modeli'nin bu seviyede daha performanslı çalıştığı ve daha doğru sonuçlar ürettiği söylenebilir. Böylece ikinci araştırma sorusu olan, model performanslarının ne olduğu ve birbirine kıyasla hangi durumda hangi modelin daha iyi sonuçlar ürettiği konusu açıklığa kavuşmuştur.

Son araştırma sorusunda, cevabı aranan tahminleme sürecinde ne gibi öğretiler elde edildiği sorusu için, çalışmalar esnasında karşılaşılan zorluk ve engellerden bahsedilerek tahminleme gerçekleştirecek kişi ve kurumlara bu kapsamda ne yapmaları gerektiği anlatılmıştır. Bu çalışmadan çıkarılan en önemli öğreti, hata ve hata yoğunluğu dağılımlarının doğasının iyi bir şekilde gözlemlenerek, eksik verilerin tanımlanması ve nasıl üstesinden gelinmesi gerektiği planlanarak kendi durumumuza en uygun tahminleme modelinin seçilmesi gerektiğidir. Hataların ortaya çıktığı ve düzeltildiği esnadaki verilerin toplanması sürecinin iyi bir şekilde değerlendirilmesi gerekmektedir. İteratif olarak geliştirilen projelerde, bir iterasyonda ortaya çıkan hataların sonraki iterasyonlarda çözülmesi olası bir durumdur. Bu nedenle, hataların ne zaman ortaya çıktığı ve ne zaman çözüldüğü bilgilerinin saklanması önemlidir. Bir diğer önemli konu, proje planlama verilerinin proje hata verileriyle tutarlı ve paralel bir şekilde saklanıyor olması gerektiğidir. Bu nedenle hataların kaydedildiği ve yönetildiği aracın, proje yönetim aracı ile entegre bir şekilde çalışması gerekmektedir.

Bu çalışma kapsamında, aynı projeye ait benzer bağlama sahip modüller üzerinde çalışmalar gerçekleştirilerek bir tahminleme süreci yönetilmiştir. Kullanılan modüller benzer bağlama sahip olsa da modül nitelikleri detayda dikkate alınmamıştır. Bu nedenle gelecek çalışmalarda, modül ve projelerin niteliklerine bağlı olarak başka projeler ve modüller üzerinde hata yoğunluklarını analiz eden çalışmalar gerçekleştirilebilir. Ayrıca, ilgili verilerin başka tahminleme modelleri kullanılarak analiz edilmesi veya uygun modellerin seçilmesi sürecinin, belli şartların değerlendirilmesi yoluyla (örneğin, model bazında kural tabanı kurarak) genişletilmesi sağlanabilir. Bunun için bir araç geliştirerek tahminleme süreci otomize edilebilir. Ek olarak, hata yoğunlukları yerine hata düzeltme efor bilgilerinin tahminlenmesi çalışmaları ile proje planının tahminlenmesi gerçekleştirilebilir.

KAYNAKLAR

[1] Standish Group, "Standish group chaos," 2009. [Online]. Available: http://www.standishgroup.com. [Accessed 28 Ekim 2013].

[2] C. Jones, "Software project management practices: Failure versus success," *CrossTalk: The Journal of Defense Software Engineering,* vol. 17, 2004.

[3] C. Jones, "Social and technical reasons for software project failures," *STSC Cr0ssTalk June,* 2006.

[4] S. A. Slaughter, D. E. Harter and M. S. Krishnan, "Evaluating the cost of software quality," *Communications of the ACM,* pp. 67-73, 1998.

[5] D. Zhang, "Applying machine learning algorithms in software development," *The Proceedings of 2000 Monterey Workshop on Modeling Software System Structures,* pp. 275-285, 2000.

[6] N. E. Fenton and M. Neil, "A critique of software defect prediction models," *Software Engineering, IEEE Transactions on,* vol. 25, no. 5, pp. 675-689, 1999.

[7] T. M. Mitchell, Machine Learning, New York, NY, USA: McGrawHill, 1997.

[8] F. V. Jensen, An introduction to Bayesian networks, London: UCL press, 1996.

[9] IEEE, "IEEE Std. 1633, IEEE Recommended Practice in Software Reliability," IEEE, Los Alamitos, 2008.

[10] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert systems with applications,* vol. 36, no. 4, pp. 7346-7354, 2009.

[11] D. Wahyudin, R. Ramler and S. Biffl, "A framework for defect prediction in specific software project contexts," in *Software Engineering Techniques*, Springer, 2011, pp. 261-274.

[12] A. G. Koru and H. Liu, "Building effective defect-prediction models in practice," *Software, IEEE,* vol. 22, no. 6, pp. 23-29, 2005.

[13] R. Moser, W. Pedrycz and G. Succi, "A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction," *Software Engineering, 2008. ICSE'08. ACM/IEEE 30th International*

Conference on, pp. 181-190, 2008.

[14] M. Kutner, C. Nachtsheim and J. Neter, Applied Linear Regression Models, New York: McGraw Hill Education, 2004.

[15] D. Wahyudin, A. Schatten, D. Winkler, A. M. Tjoa and S. Biffl, "Defect Prediction using Combined Product and Project Metrics-A Case Study from the Open Source" Apache" MyFaces Project Family," *Software Engineering and Advanced Applications, 2008. SEAA'08. 34th Euromicro Conference,* pp. 207-215, 2008.

[16] T. Gu, S. Kim and J. Baik, "Adaptive practice on software reliability based on IEEE Std. 1633 in frequent requirement modifications," in *Computer and Information Science 2010*, Springer, 2010, pp. 1-11.

[17] W. W. Royce, "Managing the development of large software systems," *proceedings of IEEE WESCON,* vol. 26, no. 8, 1970.

[18] I. Sommerville, Software Engineering (7th Edition), Pearson Education Ltd., 2004.

[19] P. McBreen, Software Craftsmanship: The New Imperative (1st Edition), Boston: Addison-Wesley Professional, 2001.

[20] C. Larman and V. R. Basili, "Iterative and incremental development: A brief history," *IEEE Computer Society,* vol. 36, no. 6, pp. 47-56, 2003.

[21] J. D. Powell and J. Spanguolo, "Modeling defect trends for iterative development," 2003.

[22] P. Abrahamsson, R. Moser, W. Pedrycz, A. Sillitti and G. Succi, "Effort prediction in iterative software development processes--incremental versus global prediction models," in *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, IEEE, 2007, pp. 344-353.

[23] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering,* vol. 14, no. 2, pp. 131-164, 2009.

[24] A. M. Vladu, S. S. Iliescu and I. Fagarasan, "Product defect prediction model," in *Applied Computational Intelligence and Informatics (SACI), 2011 6th IEEE International Symposium on*, IEEE, 2011, pp. 499-504.

[25] A. Tarhan and O. Demirors, "Investigating the effect of variations in the test development process: a case from a safety-critical system," *Software Quality*

Journal, vol. 19, no. 4, pp. 615-642, 2011.

[26] L. Qian, Q. Yao and T. M. Khoshgoftaar, "Dynamic Two-phase Truncated Rayleigh Model for Release Date Prediction of software," *Journal of Software Engineering & Applications,* vol. 3, no. 6, 2010.

[27] M. Lyu, "Software reliability: to use or not to use?," in *Software Reliability Engineering, 1994. Proceedings., 5th International Symposium on*, IEEE, 1994, pp. 66-73.

[28] T. M. Khoshgoftaar and N. Seliya, "Fault prediction modeling for software quality estimation: Comparing commonly used techniques," *Empirical Software Engineering,* vol. 8, no. 3, pp. 255-283, 2003.

[29] T. M. Khoshgoftaar and N. Seliya, "Comparative assessment of software quality classification techniques: An empirical case study," *Empirical Software Engineering,* vol. 9, no. 3, pp. 229-257, 2004.

[30] M. R. Lyu, Handbook of Software Reliability Engineering, New York: IEEE Computer Society Press and McGraw-Hill, 1996.

[31] P. Lakey and A. Neufelder, "System and software reliability assurance notebook," *Rome Laboratory,* 1997.

[32] J. D. Musa, Software Reliability Engineering: More Reliable Software Faster and Cheaper (2nd Edition), AuthorHouse, 2004.

[33] I. C. Society, "IEEE Std. 982-2, IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software," IEEE, Los Alamitos, 1988.

[34] M. Kline, "Software and hardware R&M: What are the differences?," in *Proc. Ann. Reliability and Maintability Symp.*, 1980, pp. 179-183.

[35] Lipow and M. L. Shooman, "Software Reliability," in *Consolidated Lecture Notes, Tutorial Sessions, Annual Reliability and Mantability Symposium*, 1986.

[36] S. H. Kan, Metrics and Models in Software Quality Engineering (2nd Edition), Boston, MA, USA: Addison-Wesley Professional, 2002.

[37] L. Laird, "In Praise of Defects," Stevens Institute of Technology, 2006. [Online]. Available: http://www.njspin.org/present/Linda%20Laird%20March%202005.pdf. [Accessed 20 Eylül 2013].

[38] D. C. Montgomery, E. A. Peck and G. Vining, Introduction to Linear Regression Analysis (5th Edition), Wiley, 2012.

[39] J. F. Kenney and E. S. Keeping, "Linear Regression and Correlation," in *Mathematics of Statistics, Pt. 1, (3rd ed.)*, NJ, Princeton, 1962, pp. 252-285.

[40] V. R. Basili, G. Caldiera and H. D. Rombach, "Goal Question Metric Paradigm," *Encyclopedia of Software Engineering,* vol. 1, pp. 528-532, 1994.

[41] I. 15939, "Software Engineering-Software Measurement Process," ISO/IEC 15939, 2002.

[42] A. Tarhan and O. Demirors, "Apply Quantitative Management Now," *Software, IEEE,* vol. 29, no. 3, pp. 77-85, 2012.

[43] N. Nagappan, T. Ball and A. Zeller, "Mining metrics to predict component failures," in *Proceedings of the 28th international conference on Software engineering*, New York, ACM, 2006, pp. 452-461.

[44] B. Kitchenham, C. Kutay, R. Jeffery and C. Connaughton, "Lessons learnt from the analysis of large-scale corporate databases," in *Proceedings of the 28th international conference on Software engineering*, ACM, 2006, pp. 439-444.

[45] R. Ramler and K. Wolfmaier, "Issues and effort in integrating data from heterogeneous software repositories and corporate databases," in *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, ACM, 2008, pp. 330-332.

[46] P. L. Li, J. Herbsleb, M. Shaw and B. Robinson, "Experiences and results from initiating field defect prediction and product test prioritization efforts at ABB Inc.," in *Proceedings of the 28th international conference on Software engineering*, ACM, 2006, pp. 413-422.

[47] M. Bäumer, P. Seidler, R. Torkar, R. Feldt, P. Tomaszewski and L.-O. Damm, "Predicting fault inflow in highly iterative software development processes: an industrial evaluation," in *Supplementary CD-ROM Proceedings of the 19th IEEE International Symposium on Software Reliability Engineering: Industry Track*, 2008.

[48] S. G. MacDonell and M. J. Shepperd, "Using prior-phase effort records for re-estimation during software projects," in *Software Metrics Symposium, 2003. Proceedings. Ninth International*, IEEE, 2003, pp. 73-86.

[49] Q. Hu, R. Peng, M. Xie, S. H. Ng and G. Levitin, "Software reliability

modelling and optimization for multi-release software development processes," in *Industrial Engineering and Engineering Management (IEEM), 2011 IEEE International Conference on*, IEEE, 2011, pp. 1534-1538.

[50] P. Jalote and N. Agrawal, "Using defect analysis feedback for improving quality and productivity in iterative software development," in *Information and Communications Technology, 2005. Enabling Technologies for the New Knowledge Society: ITI 3rd International Conference on*, IEEE, 2005, pp. 703-713.

[51] M. Thangarajan and B. Biswas, "Mathematical Model for Defect Prediction across Software Development Life Cycle," 2000. [Online]. Available: http://www.qaiindia.com/Conferences/SEPG2000/index.html.

[52] P. Kapur, D. Goswami, A. Bardhan and O. Singh, "Flexible software reliability growth model with testing effort dependent learning process," *Applied Mathematical Modelling,* vol. 32, no. 7, pp. 1298-1307, 2008.

[53] H. Pham, System Software Reliability, Springer, 2006.

[54] A. L. Goel and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," *IEEE Transactions on Reliability,* vol. 28, no. 3, pp. 206-211, 1979.

[55] M. Xie, G. Hong and C. Wohlin, "Software reliability prediction incorporating information from a similar project," *Journal of Systems and Software,* vol. 49, no. 1, pp. 43-48, 1999.

[56] S. Ikemoto, T. Dohi and H. Okamura, "Estimating Software Reliability with Static Project Data in Incremental Development Processes," in *Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on*, IEEE, 2013, pp. 219-224.

[57] Y. Hong, J. Baik, I.-Y. Ko and H.-J. Choi, "A value-added predictive defect type distribution model based on project characteristics," in *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*, IEEE, 2008, pp. 469-474.

[58] R. Madachy, B. Boehm and D. Houston, "Modeling Software Defect Dynamics.," 2010. [Online]. Available: http://http://csse.usc.edu/csse/TECHRPTS/2010/usc-csse-2010-509/usc-csse-2010-509.pdf. [Accessed 1 Kasım 2013].

[59] K. Srinivasan and D. Fisher, "Machine learning approaches to estimating

software development effort," *Software Engineering, IEEE Transactions on,* vol. 21, no. 2, pp. 126-137, 1995.

[60] G. D. Boetticher, "Using machine learning to predict project effort: Empirical case studies in data-starved domains," in *1st International workshop on model-based requirements engineering*, Citeseer, 2001, pp. 17-24.

[61] C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Shepperd and S. Webster, "An investigation of machine learning based prediction systems," *Journal of Systems and Software,* vol. 53, no. 1, pp. 23-29, 2000.

[62] P. C. Pendharkar, G. H. Subramanian and J. A. Rodger, "A probabilistic model for predicting software development effort," *Software Engineering, IEEE Transactions on,* vol. 31, no. 7, pp. 615-624, 2005.

[63] K. Jinzenji, T. Hoshino, L. Williams and K. Takahashi, "An experience report for software quality evaluation in highly iterative development methodology using traditional metrics," in *Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on*, IEEE, 2013, pp. 310-319.

[64] T. Fujii, T. Dohi and T. Fujiwara, "Towards quantitative software reliability assessment in incremental development processes," in *Proceedings of the 33rd International Conference on Software Engineering*, ACM, 2011, pp. 41-50.

[65] I. S. ISO/IEC, "ISO/IEC 9126, Software Engineering-Product Quality: External Metrics," 2000.

[66] "Jenkins," Hudson Labs, [Online]. Available: http://jenkins-ci.org/. [Accessed Haziran 2014].

[67] "JIRA," Atlassian, [Online]. Available: https://www.atlassian.com/software/jira. [Accessed Haziran 2014].

[68] "Subversion," Apache, [Online]. Available: http://subversion.apache.org/. [Accessed Haziran 2014].

[69] "Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE)," Eutmecal, 2011. [Online]. Available: http://www.eumetcal.org/resources/ukmeteocal/verification/www/english/msg/ver_cont_var/uos3/uos3_ko1.htm. [Accessed Kasım 2013].

[70] B. A. Kitchenham, L. M. Pickard, S. G. MacDonell and M. J. Shepperd, "What accuracy statistics really measure [software estimation]," in *Software, IEE Proceedings*, vol. 148, IET, 2001, pp. 81-85.

[71] S. D. Conte, H. E. Dunsmore and V. Y. Shen, Software engineering metrics and models, Menlo Park, Calif.: Benjamin/Cummings Pub. Co, 1986.

[72] R. K. Yin, Case Study Research: Design and Methods (4th ed.), California: SAGE Publications, 2009.

ÖZGEÇMİŞ

Kimlik Bilgileri

Adı Soyadı	: Anıl AYDIN
Doğum Yeri	: Kütahya
Medeni Hali	: Bekar
E-posta	: anil.aydin@hacettepe.edu.tr
Adresi	: Çiğdem Mah. 1584. Sok. No:13/35 Çankaya/ANKARA

Eğitim

Lise	: 2001 – 2005 Aydın Süleyman Demirel Anadolu Lisesi
Lisans	: 2005 – 2010 Hacettepe Üniversitesi Bilgisayar Mühendisliği
Yüksek Lisans	: 2011 – 2014 Hacettepe Üniversitesi Bilgisayar Mühendisliği

Yabancı Dil ve Düzeyi

İngilizce – İyi Seviyede

İş Deneyimi

2010 Temmuz - Halen: Araştırmacı, TÜBİTAK-BİLGEM - Yazılım Teknolojileri Araştırma Enstitüsü(YTE)

Deneyim Alanları

Veritabanı(SQL, PL-SQL) ve Web tabanlı yazılım geliştirme, yazılım test konularında bilgi sahibi, Java/J2EE, EJB, Hibernate, JPA, XML, Adobe Flex, SVN.

Tezden Üretilmiş Projeler ve Bütçesi

Yok

Tezden Üretilmiş Yayınlar

A. Aydın, A. Tarhan. Investigating Defect Prediction Models for Iterative Software Development When Phase Data is not Recorded: Lessons Learned, in proceedings of the 9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), Lisbon, Portugal, 2014.

Tezden Üretilmiş Tebliğ ve/veya Poster Sunumu ile Katıldığı Toplantılar

Yok