WEB YAZILIM PROJELERİNDE HOMOJEN OLMAYAN POİSSON SÜRECİ YAZILIM GÜVENİLİRLİK MODELLERİNİN KARŞILAŞTIRILMASI

COMPARISON OF NONHOMOGENEOUS POISSON PROCESS SOFTWARE RELIABILITY MODELS AT WEB SOFTWARE PROJECTS

RABİA BURCU KARAÖMER

Öğr. Gör. Dr. Oumout CHOUSEINOGLOU
Tez Danışmanı

Hacettepe Üniversitesi

Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin

Endüstri Mühendisliği Anabilim Dalı için Öngördüğü

YÜKSEK LİSANS TEZİ olarak hazırlanmıştır.

2015

RABİA BURCU KARAÖMER'in hazırladığı "**Web Yazılım Projelerinde Homojen Olmayan Poisson Süreci Yazılım Güvenilirlik Modellerinin Karşılaştırılması**" adlı bu çalışma aşağıdaki jüri tarafından **ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI**'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Prof. Dr. Timur KARAÇAY
Başkan ……

Öğr. Gör. Dr. Oumout CHOUSEINOGLOU
Danışman …….

Doç. Dr. Özlem Müge TESTİK
Üye …….

Yrd. Doç. Dr. Ayça TARHAN
Üye …….

Yrd. Doç. Dr. Banu YÜKSEL ÖZKAYA
Üye …….

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

Prof. Dr. Fatma SEVİN DÜZ
Fen Bilimleri Enstitüsü Müdürü

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 değişiklik yapmadığımı,
- Ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

12/01/2016

RABİA BURCU KARAÖMER

ÖZET

WEB YAZILIM PROJELERİNDE HOMOJEN OLMAYAN POİSSON SÜRECİ YAZILIM GÜVENİLİRLİK MODELLERİNİN KARŞILAŞTIRILMASI

Rabia Burcu KARAÖMER

Yüksek Lisans, Endüstri Mühendisliği Bölümü

Tez Danışmanı: Öğr. Gör. Dr. Oumout CHOUSEINOGLOU

Aralık 2015, 84 sayfa

Yazılım güvenilirliği, proje başarısını doğrudan etkileyen önemli kalite faktörlerinden biridir. Yazılım güvenilirliğinin modellenmesi ile bir projenin ne kadar zaman sonra ve ne kadar efor sonucunda kullanıcıya sunulabileceği tahmin edilebilir. Bu da proje kaynak ve takvim planlamasında yardımcı olabilmektedir. Bu amaçla yazılım güvenilirlik modelleri yazılımların olgunluklarını ölçmede sıklıkla kullanılmaktadır. Literatürde yazılım güvenilirlik modellerinin karşılaştırılmasına yönelik birçok çalışma bulunmasına rağmen yazılım türünü dikkate alıp bu kapsamda yazılım güvenilirlik modellerinin karşılaştırılmasının yapılmasına ihtiyaç vardır. Bu çalışma, yazılım güvenilirliğini ölçmek için kullanılan modellerin web yazılımlarındaki performanslarının karşılaştırılmasını hedeflemektedir. Bu amaçla ALTAIR Savunma ve Yazılım Teknolojileri A.Ş. firmasının dört ayrı web yazılım projesinde tutulan olay kayıtları kullanılarak altı ayrı yazılım güvenilirlik modeli karşılaştırılmıştır. Karşılaştırmada kullanılan modeller; Üstel Homojen Olmayan Poisson Süreci Goel Okumoto, Musa Üstel, Büklümlü S Şekilli Homojen Olmayan Poisson Süreci,

Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci, Yamada ve Pham-Nordmann-Zhang Kesin Olmayan Hata Tespiti (PNZ-IFD) modelleridir.

Yazılım güvenilirlik modellerinin uygulanması sırasında en çok olabilirlik tahmin yöntemi kullanılarak hata kayıt verilerine uygun model parametreleri, her bir model ve her bir proje için tahmin edilmiştir. En çok olabilirlik yöntemi ile model parametrelerinin tahmin edilmesi sırasında, elde bulunan hata kayıt verilerinin %100'ünün, %70'inin ve %50'sinin kullanılması durumları olmak üzere üç durum ayrı ayrı incelenmiş ve sonuçları değerlendirilmiştir.

Tahmin edilen parametrelerle oluşturulan modellerin projelere uygunluğu hata kareler ortalaması (Mean Square Error, MSE), ortalama bağıl hata (Mean Magnitude Relative Error, MMRE), yüzde bağıl hata sapması (Percentage Relative Error Deviation, PRED) ve dengeli tahmini bağıl hata (Average Balanced Predicted Relative Error, A.BPRE) ölçümleri kullanılarak hesaplanmıştır. Her bir model için dört proje, üç durum (%100, %70, %50) ve dört ölçüm (MSE, MMRE, PRED, A.BPRE) sonucuna göre toplamda 48 farklı ölçüm alınmıştır. Bu 48 ölçüm içerisinden her bir ölçüm için en yüksek başarıya sahip model seçilmiş ve modeller buna göre sıralanmıştır. Çalışma sonucunda Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci yazılım güvenilirlik modeli 13 durum ile en çok durumda başarılı model olmuştur. Ancak Yamada ve Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modellerinin birbirlerine benzer hata tahminleri yapıp birbirlerine yakınsadığı düşünülmektedir. Bu nedenle bu iki modelin toplamda 23 durum ile en çok durumda en iyi sonucu veren modeller olması ile kullanılan diğer yazılım güvenilirlik modellerine göre daha iyi modelleme yapacağı düşünülmektedir.

Anahtar Kelimeler: yazılım güvenilirlik modelleri, bozulma oranı, web yazılımları, homojen olmayan Poisson süreçleri

ABSTRACT

COMPARISON OF NONHOMOGENEOUS POİSSON PROCESS SOFTWARE RELIABILITY MODELS AT WEB SOFTWARE PROJECTS

Rabia Burcu KARAÖMER

Master of Science, Department of Industrial Engineering

Supervisor: Instructor Dr. Oumout CHOUSEINOGLOU

December 2015, 84 pages

Software Reliability is an important quality factor that effects projects success. By modelling software reliability, it can be estimated when and with how much effort a project can be deployed. Consequently this can contribute to the resource and schedule planning of a project. Therefore, software reliability models are frequently used for measuring the maturity of a software. A number of works exist in the literature that compare software reliability models in terms of their modelling performance, however there is a need of evaluating these reliability models by taking into account software project types. This study aims to compare the performance of software reliability models with respect to web based software projects. In accordance to this purpose, six different software reliability models, namely Exponential Non-Homogenous Poisson Process Goel Okumoto, Musa Exponential, Inflected S Shape Non-Homogenous Poisson Process, Delayed S

Shape Non-Homogenous Poisson Process, Yamada Exponential and Pham-Nordmann-Zhang Imperfect Fault Detection (PNZ-IFD), are evaluated by using the event records of four projects developed by the ALTAIR Defence and Software Technologies Company.

In the application process of the aforementioned software reliability models the maximum likelihood method is used to estimate the parameters for each model and project. 100%, 70% and 50% of the recorded data is used as input to the maximum likelihood method and these three cases are investigated and commented separately in the research.

The goodness of fit and the predictive validity of the models to the project data is tested by calculating Mean Square Error (MSE), Mean Magnitude Relative Error (MMRE), Percentage Relative Error Deviation (PRED) and Average Balanced Predicted Relative Error (A.BPRE) measures. For each model 48 separate cases which are combinations of the three situations (100%, 70% and 50%), four projects and four measures, are investigated. For each case the best model is selected and all models are sorted by their ranks. At the end of the study, it is shown that the Delayed S Shape Non-Homogenous Poisson Process model has displayed the best rankings among the alternatives. However, it is understood that the Exponential Non-Homogenous Poisson Process Goel Okumoto and Yamada models give identical responses and that these two models converge to each other with respect to the project data that has been used. Combined these two models obtain the highest ranking scores and it is concluded that these two models perform better than the other models with respect to web based software.

Keywords: software reliability models, failure rate, web based software, nonhomogeneous Poisson process

TEŞEKKÜR

Bu çalışmanın oluşumundan sonuçlanmasına kadar her aşamasında, ilgi, destek ve bilgisini esirgemeyen, tez danışmanım ve değerli hocam sayın Öğr. Gör. Dr.Oumout Chouseinoglou'na

Tez çalışmam boyunca çok büyük sabır gösteren, sevgisini, yardımını, engin bilgisini, desteğini esirgemeyen, her koşulda yanımda olduğunu hissettiren sevgili eşim Yük. Müh. Ömer Burak Karaömer'e

Çalışmam süresince gösterdikleri destekleri için ALTAIR Savunma ve Yazılım Teknolojileri A.Ş. ve desteğini ve bilgisini esirgemeyen değerli arkadaşım Mehmet Özçelik'e,

Tez çalışmam boyunca, yardımlarını ve desteklerini esirgemeyen sevgili dostlarım Yük. Gıda Müh. Dilara Şen'e, Gıda Müh. Dr. Yaşin Şen'e,

Çalışmam süresince, hiçbir sorumu yanıtsız bırakmayan ve desteğini esirgemeyen değerli arkadaşım Yük. Müh. Erdi Daşdemir'e

En önemlisi hayatım boyunca attığım her adımda destekleriyle yanımda olan, en zor şartlarda bile her türlü fedakârlığı gösteren sevgili aileme teşekkür ederim.

İÇİNDEKİLER

Sayfa

ÇİZELGELER

ŞEKİLLER

SİMGELER VE KISALTMALAR

Simgeler

$\sum$	Toplam Sembolü
Π	Çarpım Sembolü
μ	Ortalama
a	Başlangıç Hata Miktarı
b	Hata Tespit Etme Oranı
n	Örneklem Büyüklüğü

Kısaltmalar

A.BPRE	Average Balanced Predicted Relative Error (Ortalama Dengeli Tahmini Bağıl Hata)
AR-GE	Araştırma ve Geliştirme
AS	Araştırma Sorusu
BPRE	Balanced Predicted Relative Error (Dengeli Tahmini Bağıl Hata)
COTS	Commercial Off The Shelf (Kullanıma Hazır Ürün)
HTML	Hypertext Markup Language (Hiper Metin İşaretleme Dili)
ISO	International Organization for Standardization (Uluslararası Standartlar Teşkilatı)
DCOM	Distributed Component Object Model (Dağıtılmış Bileşen Nesne Modeli)
MMRE	Mean Magnitude Relative Error (Ortalama Bağıl Hata)
MSE	Mean Square Error (Hata Kareler Ortalaması)
OLE	Object Linking and Embedding (Nesne Bağlama Ve Gömme)
PRED	Percentage Relative Error Deviation (Yüzde Bağıl Hata Sapması)
SGML	Standard Generalized Markup Language (Standart Genelleştirilmiş Biçimlendirme Dili)

1. GİRİŞ

Günümüzde bilgisayar sistemleri ile hemen herkes doğrudan ya da dolaylı olarak etkileşim içerisindedir. Günlük hayatta birçok faaliyette bilgisayarları kullanmaktayız ve bu faaliyetlerin başarılı bir şekilde sürmesi, bilgisayarlara ve onun en önemli parçasından biri olan yazılımlara bağlıdır. Yazılım ürünleri, başta finans, haberleşme ve ulaşım sistemleri, enerji, sağlık, askeri ve kamu alanları olmak üzere pek çok alanda hayatımıza girmiş ve belirleyici bir konum kazanmıştır. Bu durum yazılıma olan bağımlılığı artırmakta ve yazılımın sürdürülmesini gerekli kılmaktadır [1]. Yazılıma olan bu bağımlılık nedeni ile yazılımların güvenilir bir şekilde çalışmasına olan ihtiyaç da artmaktadır. Fakat günümüzde yazılım projelerinin üçte ikisinde proje bütçesi veya takvimi aşılmaktadır ya da projeler tamamen başarısızlıklarla sonuçlanmaktadır [2]. Her başarısızlık firmaya olan güveni ve firmanın rekabet gücünü azaltmaktadır. Firmalar projelerin başarılı olabilmeleri için projelerin gidişatını takip etmeli, oluşabilecek riskleri belirlemeli ve buna göre gerekli önleyici faaliyetleri yerine getirmelidir. Bu kapsamda projelerin geleceği hakkında tahminde bulunulması projelerin önemli süreçlerinden bir tanesidir.

Yazılım güvenilirliği bir projenin olgunluğunu ölçmede kullanılabilecek önemli kalite faktörlerindendir. Bir yazılımın ne kadar olgun olduğunun, içerisinde ne kadar hata barındırdığının ve ne zaman kullanıma sunulabileceğinin tahmin edilebilmesi, proje kaynaklarının planlanması, paylaştırılması ve takvimin belirlenmesi gibi konularda proje yöneticilerine önemli avantaj sağlayabilmektedir. Bir yazılımın güvenilirliği o yazılımın belirli şartlar altında belirli bir süre bozulma yaşamadan çalışması olasılığıdır. Yazılım güvenilirliğinin modellenmesi ise hata kayıtlarının davranışının modellenerek yazılımın belirli bir süre bozulma yaşamadan çalışması olasılığının tahmin edilmesidir [3].

Endüstride gömülü yazılım, uygulama yazılımları ve web yazılımları gibi birbirinden farklı disiplinler içeren türlerde yazılımlar bulunmaktadır. Web yazılımları kullanılan teknolojiler, kalite faktörleri, kullanılabilirlik, erişilebilirlik, performans gibi birçok alanda geleneksel yazılımlara göre farklılıklar içermektedir. Bununla birlikte web yazılımları her an müşterinin kullanımına açık olmaları nedeniyle yüksek güvenilirlik gerektiren yazılımlardır.

Kaliteli bir yazılım, müşteri beklentilerini yerine getirmelidir. Müşteri beklentilerini yerine getiremeyen bir yazılım, zamanla proje maliyetlerinin artmasına ve firmanın başarısızlığına neden olacaktır. Bu durum daha kaliteli yazılım ürünlerine olan gereksinimi artırmaktadır. Yazılım güvenilirliği, yazılım projelerinde kullanılan kalite faktörleri olan kullanılabilirlik, erişilebilirlik ve performans gibi faktörlere göre benzer öneme sahiptir. Bu nedenle güvenilirlik konusunda yapılan çalışmalar sonucunda çok sayıda yazılım güvenilirlik modeli geliştirilmiştir. Fakat hangi tür modelin hangi yazılım türlerinde uygulanabileceği hakkında yeterli bilgi bulunmamaktadır [1]. Bir yazılım güvenilirlik modeli her tür yazılım için uygun değildir. Aynı zamanda bir yazılım türü için de her model uygun sonuç vermeyebilir. Bu çalışmada web yazılımlarının özellikleri incelenerek, web yazılımlarına uygulanabilecek modellerin performanslarının karşılaştırmalarının yapılması amaçlanmaktadır. Her yazılım güvenilirlik modeli kendine has bazı varsayımlara dayanır ve bu varsayımlar modelin davranışını belirler. Kullanılacak yazılım güvenilirlik modelinin seçilmesinde bu varsayımlara dikkat edilmelidir. Ayrıca her modelin uygulanabilmesi için kendine özgü bazı verilere ihtiyacı vardır. Elde bulunan veriler ile modellerin uygulanıp uygulanamayacağı kontrol edilmeli ve verilere en uygun modeller kullanılmalıdır.

Bu çalışma kapsamında yazılım güvenilirliğinin modellenmesinde kullanılan altı farklı modelin web yazılımlarındaki performanslarını karşılaştırmak için ALTAIR Savunma ve Yazılım teknolojileri A.Ş. firmasına ait dört farklı projenin hata kayıt verileri elde edilmiş ve yazılım güvenilirlik modellerinin bu projelerdeki uygulaması yapılmıştır.

Projelerde hataların oluşma zamanı kayıt altına alınmıştır. Bu kayıtlardan her bir proje için kümülatif hata sayıları elde edilmiş ve bu verilere uygun güvenilirlik modelleri ile modellenmiştir. Bu kapsamda Homojen Olmayan Poisson Süreci modellerinin eldeki veriye en uygun modeller olduğuna karar verilmiş ve literatürdeki altı ayrı Homojen Olmayan Poisson Süreci modelinin uygulaması yapılmıştır. Seçilen güvenilirlik modellerinin ortalama değer fonksiyonlarındaki parametreler en çok olabilirlik tahmin yöntemi ile tahmin edilmiş ve hata kayıt verilerine uygun modeller elde edilmiştir. Parametre tahminleri üç farklı durum için yapılmıştır ve her bir durum ayrı ayrı incelenmiştir. İlk durumda her bir proje için hata kayıt verilerinin %100'ü kullanılarak parametre tahmini yapılmıştır. İkinci ve üçüncü durumlarda ise her bir projenin hata kayıt verilerinin sırası ile ilk %70'i ve ilk %50'si kullanılarak

parametre tahminleri yapılmıştır. Parametre tahminleri sonucu elde edilen modellemelerin performansını ölçmek ve modelleri birbirleri ile karşılaştırmak amacı ile hata kareler ortalaması (Mean Square Error, MSE), ortalama bağıl hata (Mean Magnitude Relative Error, MMRE), yüzde bağıl hata sapması (Percentage Relative Error Deviation, PRED) ve dengeli tahmini bağıl hata (Balanced Predicted Relative Error, A.BPRE) değerleri hesaplanmıştır. Tüm projelerin hata kayıt verilerinin %100'ünün parametre tahmininde kullanıldığı durumda, yine aynı veriler ile modellerin performansları ölçülmüştür. İkinci ve üçüncü durumlarda ise hata kayıt verilerinin sırasıyla kalan %30 ve %50'lik kısmı modellerin performans ölçümünde kullanılmıştır.

Bu çalışmada aşağıdaki araştırma sorularının (AS) yanıtlanması amaçlanmaktadır:

AS-1: Web yazılımlarında Homojen Olmayan Poisson Süreci yazılım güvenilirlik modelleri kullanılabilir mi?

AS-2: Homojen Olmayan Poisson Süreci yazılım güvenilirlik modellerinin web yazılımlarındaki performansları nedir ve birbirlerine kıyasla nasıldır?

AS-3: Projelerin hata kayıtlarının %70 ve %50'sindeki veriler kullanılarak tahmin edilen parametreler projenin ileriki aşamalarındaki hataların tahmin edilmesinde ne derece başarılı olmaktadır?

Bu çalışma ile literatürde bulunan yazılım güvenilirlik modelleri özetlenmiş ve nasıl kullanılabileceği detaylı olarak anlatılmıştır. Ayrıca web yazılımlarında kullanılabilecek altı ayrı yazılım güvenilirlik modeli karşılaştırılarak uygun modelin seçiminde yöntem önerilmiştir. Gelecekte yazılım güvenilirlik modellerinin karşılaştırılması ile ilgili yapılacak çalışmalarda bu çalışmada elde edilen sonuçların yol gösterici olacağı düşünülmektedir.

Çalışmanın ilerleyen bölümleri şu şekilde organize edilmiştir: ikinci bölümde yazılım güvenilirliği ve literatürde bulunan yazılım güvenilirlik modelleri hakkında geniş bilgilere yer verilmiş ve çalışmada bahsi geçen teknik terimler ve alandaki ilgili çalışmalardan bahsedilmiştir. Üçüncü bölümde ise uygulanan metodoloji ve kullanılan veri hakkında detaylı bilgi verilmiştir. Dördüncü bölümde çalışmada incelenen altı model için elde edilen sonuçlar ve bu sonuçların yorumlarına yer verilmiştir. Beşinci bölümde ise bu çalışma kapsamında elde edilen sonuçların genel yorumlanması, çalışmanın geçerliliğini etkileyebilecek faktörler, yazılım güvenilirlik

modellerinin karşılaştırılmasına ve gelecekte yapılması önerilen çalışmalara değinilmiştir.

2. LİTERATÜR ARAŞTIRMASI

2.1. Güvenilirlik

Günümüzde gelişen teknoloji ile her gün yeni ürünler üretilmekte, yeni sistemler kurulmakta olup üreticiler rekabet ortamına ayak uydurabilmek için kaliteli ürünler üretmek zorundadırlar. Kalitenin ölçülmesinde kullanılan önemli bir faktör güvenilirliktir. Güvenilirlik, bir ürünün belli bir zaman aralığında, beklenen fonksiyonunu hatasız bir şekilde yerine getirmesi olasılığıdır [4].

Bir nesnenin güvenilirliği olasılık, rassal değişkenler, dağılım fonksiyonu, yoğunluk fonksiyonu gibi olasılıksal parametreler ile ifade edilir. Genel olarak güvenilirlik, tekrar eden denemelerde aynı sonucu vermeye eğimli deneysel çalışmaları konu alır [5].

2.2. Yazılım Güvenilirliği

Yazılım güvenilirliği, bir yazılımın belirli koşullar altında, belirli bir zaman aralığında hata vermeden çalışmasının olasılığıdır; yani yazılımın gerekli fonksiyonu belirli zaman periyodunda yerine getirmesidir [3, 6]. Yazılım güvenilirliğine ait bu tanımda güvenilirlik ile kalite arasındaki yakın ilişki vurgulanmaktadır. Güvenilirlik bir yazılımın en önemli ve ölçülebilir kalite faktörlerinden birisidir [3, 7]. Yazılım kalitesini arttırmak için yapılan tüm test, ölçüm ve analizlerin asıl amacı, yazılımın belirli süre içerisinde görevini hatasız yerine getirmesini sağlamaktır. Başka bir deyişle yazılım güvenilirliğini arttırmaktır. Yazılım piyasasında en önemli problemlerden biri, yazılımın gerekli güvenilirliği ne zaman sağlayacağı ve ne zaman yazılımın kullanıma alınacağının tam olarak öngörülememesidir [8]. Bu öngörüye de ancak yazılım hakkında ölçümler toplayıp, güvenilirlik analizleri yapılması sonucunda ulaşılır.

Yazılım güvenilirliği, kalite kontrol ve test faaliyetlerinin düzenlenmesi ve sonlandırılmasına ilişkin çalışmalarda da yol göstericidir. Böylelikle test faaliyetlerinin sonlandırılarak, ürünün pazara sürülme zamanına karar verilmesi veya pazara sürülme zamanı esas alınarak, buna uygun test planlamasının yapılması, ürünün satış sonrasındaki çalışma zamanındaki en iyi ve en kötü senaryolar hakkında ön bilgilerin elde edilmesi sonucunu doğurmaktadır ki bu açıdan elde edilecek bilgiler, yönetsel kararları önemli ölçüde desteklemektedir [1].

Yazılım geliştirilirken yazılıma bir takım hatalar yerleştirilmektedir. Bu hataların yazılımdaki miktarı ile güvenilirlik ters orantılıdır. Yazılım güvenilirliğinin ölçülmesi hata oluşumlarının incelenerek, gelecekteki hata oluşumlarının modellenmesine dayanır. Güvenilir bir yazılımdan söz edildiğinde, ürünün planlama, geliştirme, test ve bakım aşamalarında hedeflenen güvenilirlik derecesini karşıladığı anlaşılmalıdır. Yazılımın hedeflenen güvenilirlik derecesini karşılayıp karşılayamadığı, yazılım güvenilirlik modelleri ile yapılan hata oranı tahminleri ile belirlenebilmektedir [1].

Yazılım güvenilirliğinin temel kavramları olan hata, hata oranı ve hata yoğunluğu tanımları aşağıdaki gibidir [6, 9].

Hata ("Fault"): Yazılımda yer alan yanlış bir adım, süreç veya veri tanımıdır. Yazılım biriminin kendinden beklenen işlevselliği göstermesine engel bir durumdur ve genellikle tasarım değişikliği ile giderilir.

Hata Yoğunluğu ("Error Density"): Yazılımda birim başına düşen hata sayısıdır.

Bozulma ("Failure"): Yazılımın veya bileşeninin belirlenen performans gereksinimleri doğrultusunda kendinden beklenen davranışı göstermemesi durumudur.

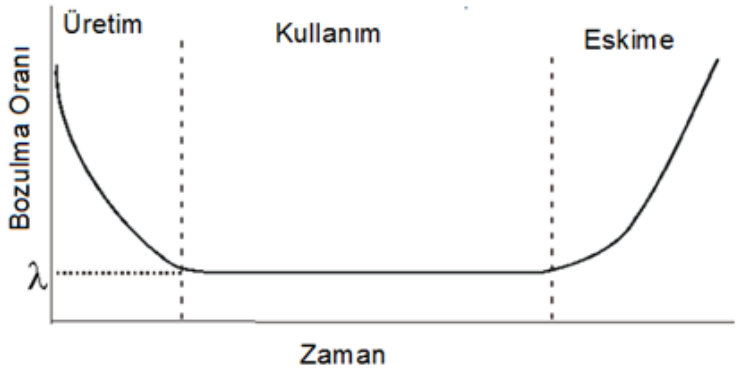
Bozulma Oranı ("Failure Rate"): Verilen bir sürede ortaya çıkan belirli bir kategori ya da önemlilik derecesine sahip bozulmaların sayısıdır.

2.3. Güvenilirlik Açısından Yazılım ve Donanım Arasındaki Farklılıklar

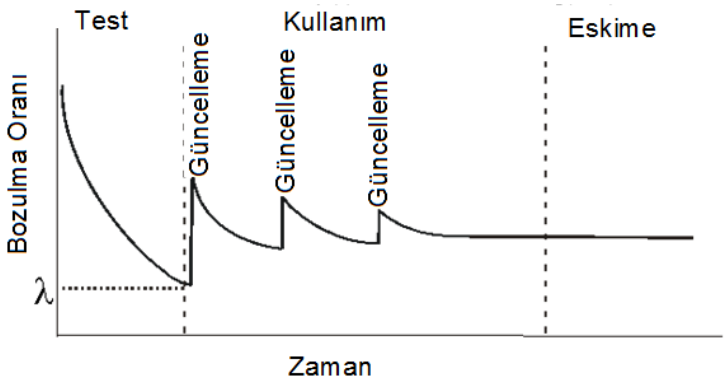
Yazılım ve donanım güvenilirlik açısından farklılıklar içermektedir. Her ne kadar güvenilirlik tanımı yazılım ve donanım için kayda değer bir farklılık içermese de, yazılım ve donanım arasındaki temel farklar, yazılım ve donanımın güvenilirliğinde farklılıklar olmasına neden olur. Yazılım ve donanımın güvenilirliğinin ölçülmesinde benzer matematiksel modellemeler kullanılabilmektedir. Ancak genellikle hata oluşumlarının incelenmesiyle geliştirilen matematiksel modellemeler, yazılım ve donanım hata oluşumunun birbirinden farklı olmasından, yazılım ve donanım için farklılık göstermektedir.

Yazılımda tespit edilen hatalar genellikle tasarımdan kaynaklı hatalar iken donanımda tespit edilen hatalar genellikle fiziksel hatalardır. Çevresel değişiklikler yazılımda yeni gereksinim, yeni beklentiye sebep olurken donanımda eskimeye sebep olur. Yazılım ürünleri üretilmez, geliştirilir yani yazılım ikinci bir yazılım ürünü

için sadece kopyalanabilir ve bu kopya ürün birinci ürün ile birebir aynı özellikleri taşır [10, 11]. Yazılımda hatanın gözlemlenmesi direkt olarak zamandan etkilenmez. Hatanın gözlemlenmesi daha çok test kapsamına bağlıdır. Donanımda ise hatanın gözlemlenmesi zamana bağlıdır ve zaman ilerledikçe donanımın eskimesine bağlı olarak hata görülme olasılığı artar [12]. Fiziksel girdilerinden yazılım güvenilirliği etkilenmez ancak donanım güvenilirliği sıcaklık, nem gibi çevre faktörlerinden etkilenir. Bir hata düzeltildiğinde artık yazılımın tüm sürüm kopyalarında bu hata düzeltilmiştir ve yazılımda bir hatanın düzeltilmesi yeni hataların oluşmasına etken olur iken donanımda bir üründe bulunan hataların düzeltilmesi diğer ürünlerde de o hatanın çıkmayacağını garanti etmez [10, 11].



Şekil 1 Donanım Güvenilirlik Eğrisi [13]



Şekil 2 Yazılım Güvenilirlik Eğrisi [13]

Şekil 1 Donanım Güvenilirlik Eğrisi [13] incelendiğinde, donanım güvenilirliğinin üç zaman diliminde farklı davrandığı görülebilir. Eğrinin ilk bölümü donanımın test edilip belirli bir güvenilirlik seviyesine getirildiği bölümdür. Bu aşamada yapılan test ve düzeltme faaliyetleri sonucu donanımda bulunan bozulma oranı gittikçe azalır ve

güvenilirlik artar. Belirli güvenilirlik seviyesine geldiğinde artık donanım kullanılabilir ürün olarak kullanıcıya sunulur. Eğrinin ikinci aşaması ürünün kullanıcı tarafından kullanıldığı zaman dilimidir. Donanımdaki bozulma oranı bir süre sabit kalır. Bu süre boyunca donanımda herhangi bir değişiklik olmadığından güvenilirlik de sabittir. Eğrideki üçüncü aşamada ise artık donanım eskimeye başlamıştır ve eskimeye bağlı olarak donanımın bozulma oranı artış gösterir. Zamanla güvenilirliği iyice azalır ve kullanılabilir olma özelliğini kaybeder.

Şekil 2 Yazılım Güvenilirlik Eğrisi [13] incelendiğinde, yazılım güvenilirliğinin donanım güvenilirliğine göre farklılıklar içerdiği görülebilir. Yazılım güvenilirlik eğrisinin ilk aşaması donanım güvenilirlik eğrisinde olduğu gibi, geliştirme ve test aşamasıdır. Bu aşamada testler ve hata düzeltici faaliyetler sonucu yazılım bozulma oranı azaltılarak, yazılımın belirli bir güvenilirlik seviyesine ulaşması sağlanır. Yazılım güvenilirlik eğrisinin ikinci aşamasını incelediğimizde, donanım güvenilirlik eğrisi ile asıl farkın bu aşamada başladığı görülebilir. Geliştirilen ürünün kullanıcı tarafından kullanıldığı bu bölümde, donanım bozulma oranı ve güvenilirliği sabit kalırken, yazılım bozulma oranı ve güvenilirliği dalgalanma göstermektedir. Bu durumun nedeni, yazılımın sürekli güncellenebilir olması özelliğindendir. Kullanım durumundaki bir yazılım, çevresel faktörlerin değişmesi, kullanıcı ihtiyacının değişmesi, gelişen teknoloji ve artan ihtiyaç gibi nedenlerle güncellenmek durumunda kalır. Yazılımın güncellenmesi durumunda, geliştirilmesindekine benzer şekilde yazılıma yeni hatalar eklenir ve yazılımın bozulma oranı artış gösterir. Tekrar eden test ve hata düzeltici faaliyetler ile bozulma oranı azaltılarak yazılım güvenilirliğinin tekrar artması sağlanır. Yazılım kullanıldığı süre boyunca güncellemeler ve devamında test ve hata düzeltici faaliyetler devam eder. Yazılım güvenilirlik eğrisindeki son aşamada ise artık yazılımın kullanımı sona ermiştir. Kullanılmayan bir yazılıma güncelleme de yapılmaz. Bu aşamada donanımdan farklı olarak zamanla eskime göstermeyen yazılımın bozulma oranı da değişmez.

2.4. Web Yazılımları ve Geleneksel Yazılım Türlerinin Karşılaştırılması

Web yazılımlarının ticari başarısı için yüksek güvenilirlik çok kritiktir. Eğer bir web yazılımı yeterince güvenilir şekilde çalışmıyor ise, başarılı olamaz. Çoğu web yazılımı yüksek rekabet şartları altında çalışmaktadır. Yüksek rekabet, kullanıcıların, yazılımdan beklentilerini, yazılımın ne oranda karşıladığına göre

seçim yapabilmesi demektir. Eğer bir web uygulaması beklentiyi karşılamıyor ise, kullanıcılar başka bir web uygulamasını kullanmayı tercih edeceklerdir [14].

Yazılımların güvenilirlik ölçümlerinin sağlıklı yapılabilmesi için yazılımın geliştirme süreci, kullanılan teknolojiler, yazılımın kullanım durumları ve benzeri birçok faktörün incelenmesi gerekmektedir. Kullanıcı ara yüzü olmayan gömülü bir yazılımla, bir web sayfasının test edilme süreci ve test durumları nasıl fark ediyor ise, güvenilirlik analizinin yapılması da aynı şekilde farklılıklar içerebilir. Benzer bir şekilde ağ bağlantısı olmayan bir uygulamanın çevresel faktörleri ile bir web yazılımının çevresel faktörlerinin aynı olması beklenmediği gibi güvenilirlik ölçümlerinin de aynı olması beklenemez.

Bu çalışma kapsamında web yazılımlarının güvenilirliğinin modellenmesi incelenirken, web yazılımları ile geleneksel yazılımlar arasındaki farklar da araştırılmıştır. Aşağıdaki başlıklarda web yazılımlarının geleneksel yazılımlardan farklılıkları anlatılmıştır [15].

2.4.1. Uygulama Karakteristikleri

Web uygulamaları birçok farklı elemanın biraraya gelmesi ile oluşur. Örneğin dağıtılmış bileşen nesne modeli (Distributed Component Object Model, DCOM), ActiveX, nesne bağlama ve gömme (Object Linking and Embedding, OLE) gibi bileşenler, yorumlanmış script dilleri, ticari kullanıma hazır (Commercial Off The Shelf, COTS) yazılımlar, multimedya dosyaları, hiper metin işaretleme dili (Hypertext Markup Language, HTML)/ standart genelleştirilmiş biçimlendirme dili (Standard Generalized Markup Language, SGML) /genişletilebilir işaretleme dili (Extensible Markup Language, XML) dosyaları, grafik resimler, HTML ve program karışımları, veri tabanları gibi elemanlardan oluşabilir. Bu bileşenler çok farklı bir şekilde entegre edilebilir. Ayrıca bileşenlerin kaynak kodları erişilemez olabilir ve başka bir bilgisayarda çalışıyor olabilir. Web uygulamaları platformdan bağımsızdır ve genellikle tüm web tarayıcılarında aynı işlevsellik ve görsellik ile çalışır. Ayrıca web uygulamalarını diğer yazılımlardan ayıran önemli bir özelliği de hipermedya paradigmasını kullanmasıdır. İçerikler hiperlink kullanılarak oluşturulan yapılarda saklanır ve sunulur. Ama geleneksel yazılımlar çoğunlukla üzerinde çalıştıkları donanıma veya işletim sistemine bağımlıdırlar. Java gibi platformlar arası kodlama

dilleri vardır ama geleneksel yazılımların çoğu tek parça halinde, tek bir işletim sisteminde çalışacak şekilde olma eğilimindedir [15, 16].

2.4.2. Kullanılan Temel Teknolojiler

Web yazılımları çok çeşitli teknoloji beraber kullanılarak geliştirilir. Örneğin işletim programlama dilleri, sunucular, betikler, web tasarım dilleri, veri tabanları ve çok daha fazlası web yazılımı geliştirirken kullanılabilir. Ayrıca üçüncü parti yazılımlar ve ara katman yazılımlarının kullanımları da gittikçe artmaktadır. Web teknolojisi o kadar hızlı değişmektedir ki, bazı yazarlar şirketlerin güncel teknolojileri takip etmesinin çok zor olabileceğini düşünmektedir [15-17].

2.4.3. Kalite Yaklaşımları

Web yazılımlarında kalite her zaman ön plandadır. Müşteri, yapılan işin kalitesi hakkında şüpheye düşerse başka firmalara yönelebilir. Kalite, müşteri ile tekrar iş yapabilmek için birinci öneme sahiptir. Bu nedenle web yazılımlarında daha geç ama daha kaliteli ("later and better"), yazılım stratejisi oldukça geçerlidir [15].

2.4.4. Geliştirme Sürecini Etkileyen Faktörler

Web yazılımının geliştirme sürecinde temel olarak üç kalite kriteri vardır. Bunlar güvenilirlik, kullanılabilirlik ve güvenliktir. Ayrıca ulaşılabilirlik, ölçeklenebilirlik, idame edilebilirlik ve yayınlanma süresi de ikincil öneme sahip kalite gereksinimleridir [15, 16].

2.4.5. Kullanılabilirlik/Erişilebilirlik

Web uygulamalarında müşteri uygulamayı heran kullanabilir. Bu yüzden herhangi bir anda bile uygulamanın kullanılamaz durumda olması çok büyük sorunlara neden olabilir [15].

2.4.6. Müşteriler

Web uygulamalarında kullanıcılar bir şirket içinde sınırlı sayıdaki belirli gruplar olabileceği gibi, herhangi bir kişi olabilir. Genellikle kullanıcılarının kim oldukları bilinen uygulamalara göre kullanıcılarının kim oldukları bilinmeyen uygulamalar daha fazladır. Bu nedenle web yazılımı geliştiricileri genellikle bilinmeyen kullanıcı sorunuyla karşı karşıyadırlar. Bu durumda yazılımın geliştirme aşamasında kullanıcıların beklentileri, gereksinimleri ve davranış biçimleri bilinemez. Ayrıca,

kullanıcının web yazılımını nasıl kullanacağını bilmemesi web yazılımları için genel bir durumdur. Yazılımı tanımaya çalışan kullanıcı, geliştirme aşamasında test edilmeyen durumları deneyebilir. Yazılımın kalite gereksinimlerini sağlamak için faklı yaklaşımlar geliştirmek gerekir. Kullanıcı bilinmediği için, başarılı yazılım için şart olan kullanıcı ara yüzlerinin, istenilen seviyede olgun olması da zorlaşmaktadır. Kullanıcıların bilinmemesi yanında kullanıcıların çok farklı ülkelerden, farklı kültürlerden olabilmesi de web yazılımlarının geliştirilmesinde zorluk yaratmaktadır. Bunlarla beraber kullanıcı sayısının bilinememesi de ayrıca sorun oluşturmaktadır [15, 17].

2.4.7. Bakım ve Güncelleme

Web yazılımları genellikle çok sık güncellenir. Bakım döngüleri günler, hatta saatler olabilir. Bununla birlikte işlevsellikleri ve içerikleri anlık olarak değişebilir, yani proje bitmesi gibi bir durum hiç olmayabilir. Geleneksel yazılımlarda ise yazılımın ne zaman yayınlanacağı daha önceden belirlenmiştir. Yazılım bakım döngüleri belirli bir plana göre yapılır ve ileride yapılacak değişiklikler için zamanlama önceden planlanmıştır [15, 18].

2.4.8. Yazılım Mimarisi ve Ağ Yapısı

Web yazılımları çoğunlukla işlemci/sunucu mimarisi ile çalışır. Kullanıcı bilgisayarındaki web tarayıcısı ile sunucu bilgisayarı birlikte iki katmanlı bir yapıdadır. Kullanıcı ve sunucu tarafındaki donanımlar işletim sistemleri ve bunlar gibi birçok şey farklı olabilir. İki katmanlı mimarinin yetersiz kaldığı birçok durumda ise üç katmanlı veya daha çok katmanlı mimariler kullanılır. Bu yapılarda iş mantığını barındıran ara katman oluşturulur ve bu katman işlemci tarafından gelen veri ve fonksiyon isteklerine cevap verirken gerekli veriyi genellikle veri tabanı olan bir alt katmandan ister. Ayrıca kullanıcı tarafındaki ağ yapısı bilinmediği için web yazılımı geliştirilirken farklı ağ yapıları da düşünülerek geliştirme yapılmalıdır [15, 17].

2.4.9. İçerilen Disiplinler

Web yazılımları çok farklı disiplinlerden insanların bir araya gelmesi ile geliştirilir. Geliştirme süreci yazılım mühendisliği, hipermedya mühendisliği, gereksinim

mühendisliği, kullanılabilirlik mühendisliği, bilgi teknolojileri, grafik tasarımcılığı, ağ yönetimi gibi alanları bir araya getirir [15, 16].

2.4.10. Yasal, Sosyal ve Etik Konular

Web yazılımlarında kullanılan içerik kolayca kopyalanabilir. Bu durum içeriğin sahibinden habersiz bir şekilde yapılabilir ve yasal, sosyal ve etik açıdan sorunlara neden olabilir. Yazılım geliştiriciler yazılımları tüm hakları gözeterek geliştirmek zorundadırlar. Web yazılımlarında kullanıcıların bilinmemesi ve sayılarının çok olması geliştirme sırasında yasal, sosyal ve etik açıdan alınması gereken önlemleri arttırmaktadır [15].

2.5. Yazılım Güvenilirlik Modelleri

İstatistiksel ve olasılıksal teorilere dayanan, matematiksel yazılım güvenilirlik modelleri, belirli zamana kadar tespit edilmiş toplam hata sayısını, yazılımın bozulma oranını ve yazılım güvenilirliğini sayısal verilerle tahmin edilmesinin tanımlanmasında genellikle kullanılan yöntemlerdir [19]. Yazılım güvenilirlik modelleri, yazılımın bilinen ya da varsayılan karakteristiklerini kullanarak yazılımın bozulma davranışını öngörür [20]. Yazılım geliştirme sürecinin test aşamasında yazılımda bulunan hataların bir kısmı tespit edilir ve yazılım yüksek bir eforla düzeltilir. Test işlemi devam ettiği sürece bulunan hatalarla, yazılımda kalan hataların sayısı gittikçe azalır. Yani hata bulma olasılığı azalır iken yazılım güvenilirliği gittikçe artar [21]. Yazılımların güvenilirliğini yazılım güvenilirlik modellerini kullanarak tahmin edebiliriz. Temel olarak iki tür yazılım güvenilirlik modeli vardır. Bunlar deterministik ve olasılıksal modellerdir [22].

2.5.1. Deterministik Modeller

Deterministik modellerin modellenmesinde programdaki operatör ve işlem gören terim sayısı kullanılır. Deterministik türdeki performans ölçümleri rasgele olayları değil, program yapısını analiz ederek elde edilir. Deterministik modellerden en iyi bilinen iki tanesi Halstead Yazılım Ölçümü ve McCabe Karmaşıklık Ölçümüdür. Halstead’s Yazılım Ölçümü programdaki hata sayısını tahmin etmek için kullanılır iken McCabe’nin Karmaşıklık Ölçümü programda kalan hata sayısının üst sınırını tahmin etmek için kullanılır [22].

2.5.1.1. Halstead Yazılım Ölçümü

Halstead Yazılım Ölçümü, yazılımın karmaşıklığını tahmin etme tekniklerinden biridir. Halstead'a göre bir bilgisayar yazılımı, operatörlerden ve işlem gören terimlerden oluşan simgeler koleksiyonu olarak düşünülebilir [23]. Halstead, yazılımda bulunan ayrı operatör ve işlem gören terim sayısını kullanarak yazılımın karmaşıklığını, büyüklüğünü, uzunluğunu ve yazılımda kalan hata sayısını tahmin etmek için matematiksel formüller üretmiştir.

2.5.1.2. McCabe'nin Karmaşıklık Ölçümü

McCabe'nin Karmaşıklık Ölçümü, yazılımın kontrol akışının grafiğine dayanan ve bu grafiğe göre yazılım karmaşıklığını ölçmeyi amaçlayan bir yöntemdir. Bu yönteme göre yazılım kontrol grafiğindeki karar noktalarının sayısı bize yazılımın karmaşıklığı hakkında bilgi verir. Bir grafiğin karmaşıklık sayısı aşağıdaki eşitlikle hesaplanabilir.

$$V(G) = e - n + 2p$$

Formülde e kenar sayısını, n köşe sayısını ve p bağlı eleman sayısını ifade eder. Karmaşıklık sayısı bir yazılımın karmaşıklığı ve yazılımın tasarımında yazılım standartlarına ve iyi yazılım uygulamalarına uyulup uyulmadığı hakkında bilgiler de verir [22].

2.5.2. Olasılıksal Modeller

Olasılıksal modeller, yazılımda hata oluşumunu ve hataların giderilmesini olasılıksal olaylar olarak ifade ederler. Olasılıksal yazılım güvenilirlik modelleri farklı gruplarda sınıflandırılabilir. En iyi bilinen olasılıksal yazılım güvenilirlik modelleri;

- Hata Kaynağı Modelleri,
- Bozulma Oranı Modelleri,
- Güvenilirlik Geliştirme Modelleri,
- Homojen Olmayan Poisson Süreci Modelleridir.

2.5.2.1. Hata Kaynağı Modelleri

Hata Kaynağı yazılım güvenilirlik modelleri çok aşamalı örnekleme teknikleri kullanarak programdaki hata sayısını tahmin eder [22]. Hata Kaynağı modellerinde, hatalar doğal ve eklenmiş hatalar olmak üzere ikiye ayrılır. Doğal hata sayısı

eklenmiş hata sayısı ve test sırasında bulunan iki türden hataların oranı ile tahmin edilir [13].

Bu tür modellerin temel yaklaşımı içerisinde bilinmeyen sayıda doğal hata bulunduran yazılıma bilinen sayıda hata eklenmesidir. Yazılım test edilir ve doğal ve eklenen hata sayısı gözlemlenir. Bu gözlem doğrultusunda yazılımdaki toplam doğal hata sayısı tahmin edilmeye çalışılır [24].

Eklenen hataların zorluk derecesinin, konumlarının ve tiplerinin karar verilmesinde yaşanan zorluklar ve tespit edilen doğal hata ile yaklaşık aynı oranda eklenmiş hata tespit edilmesinin beklenmesidir [22].

2.5.2.1.1. Mill'in Hata Kaynağı Modeli

Mill'in Hata Kaynağı Modeli, programdaki hata sayısını tahmin etmek için tasarlanmış en çok kullanılan hata kaynağı modellerinden biridir. Bu model bir miktar bilinen hatanın yazılıma eklenmesini gerektirir. Yazılım bir süre test edilir ve hatalar bulunur. Bulunan hatalar içerisindeki doğal ve eklenmiş hataların sayısından programda henüz tespit edilememiş doğal hata sayısı tahmin edilir [7, 24]. Eğer doğal ve eklenmiş hataların tespit edilme olasılıkları eşit ise r tane yok edilmiş hatadan k tanesinin doğal hata olma olasılığı Hipergeometrik dağılım gösterir.

$$P(k;N,n_1,r)=\frac{\binom{n_1}{k}\binom{N}{r-k}}{\binom{N+n_1}{r}},k=1,2,3,...,r$$

,

$N=$ Toplam doğal hata sayısı

$n_1=$ Toplam eklenmiş hata sayısı

$r=$ Hata ayıklama sırasında kaldırılan hata sayısı

$k=$ r tane kaldırılan hatadaki eklenmiş hata sayısı

$r-k=$ r tane kaldırılan hatadaki doğal hata sayısı

n_1, r ve k bilindiğinde N parametresi aşağıdaki şekilde gösterilebilir:

$$N_0 = \frac{n_1(r-k)}{k} - 1$$

$$\hat{N} = (N_0) + 1$$

Eğer N_0 bir tam sayı ise hem N_0 hem de N_{0+1}, N parametresinin en çok olabilirlik tahmin edicisidir.

Mill'in hata kaynağı modelinin birkaç dezavantajı vardır. Bunlar; modelin uygulanması ile test eforunun artması, eklenmiş hataların zorluk derecesinin, konumunun ve tipinin belirlenmesinde yaşanan zorluklar ve tespit edilen doğal ve eklenmiş hatanın yaklaşık aynı oranda tespit edilmesidir [22].

2.5.2.1.2. Cai Modeli

Cai Modeli, Mill'in hata kaynağı modelini değiştirerek yazılımı Parça0 ve Parça1 Hipergeometrik Dağılım Modeli olarak ikiye böler. Modelin uygulanabilmesi için varsayımlar aşağıdaki gibidir;

- Yazılımda N adet kalan hata sayısı var ise Parça0 (N_0), Parça1 (N_1) adet olmak üzere $(N = N_0 + N_1)$ dir.
- Her bir kalan hatanın tespit edilme olasılığı aynıdır.
- Hata tespit edildiği an kaldırılır (çözümlenir)
- Her bir zamanda sadece bir hata kaldırılır ve bu hata kaldırılır iken yeni hata tanımlanmaz
- Kaldırılan hataların sayısı n'dir [22].

2.5.2.1.3. Hipergeometrik Dağılım Modelleri

Tohma, hipergeometrik dağılıma dayanan ve test fazı başlangıcında yazılımda bulunan toplam hata sayısını tahmin eden bir model geliştirmiştir. Modele göre C_{i-1}, t_{i-1} anına kadar bulunmuş hataların kümülatif toplamı, ve N_i, t_i anında yeni bulunan hataların sayısını ifade eder. Yazılım başlangıçta m tane hata içerir. Testler girdi ve çıktılardan oluşur ve bir günde veya haftada koşulacak testlerin kümesine test durumu denir. Test durumları t_i ile gösterilir ve test durumları arasında hatalar düzeltilmez. Yani bir hata birden çok test durumunda ortaya çıkabilir. Tohma'nın

geliştirdiği modele göre W_i, t_i test durumunda ortaya çıkan hata sayısı ise, bu hataların içinde C_{i-1} tane daha önce çıkmış hatalardan bulunması olasılığı vardır. N_i sayısının hipergeometrik dağılım gösterdiğini düşünürsek, W_i tane çıkan hatanın içerisinde n_i tane yeni ortaya çıkmış hatanın olma olasılığı;

$$P(N_i = n_i) = \frac{\binom{m - C_{i-1}}{n_i}\binom{C_{i-1}}{W_i - n_i}}{\binom{m}{W_i}}$$

eşitliği ile hesaplanır [22].

2.5.2.2. Bozulma Oranı Modelleri

Bozulma Oranı Modelleri yazılımda oluşan hatalar arasındaki süreyi kullanarak yazılım güvenilirliği hakkında tahminde bulunur. Bu tür modellerde yazılımda bulunan hataların oranının zaman içerisinde değişimi incelenir. Bozulma oranı modellerinde yazılımda kalan hataların sayısı değiştikçe yazılımın bozulma oranı değişir ve bu değişim modellenir. Bozulma oranı modeli grubuna giren yazılım güvenilirlik modelleri [7, 22];

- Jelinski Moranda
- Schick Wolverton
- Jelinski Moranda Geometrik
- Moranda Geometrik Poisson
- Değiştirilmiş Schick ve Wolverton
- Hata Düzeltme Sürecinin Tamamlanmadığı Durumlarda Goel Okumoto Modeli

2.5.2.2.1. Jelinski Moranda Modeli

Jelinski Moranda modeli en eski yazılım güvenilirlik modellerinden biridir. Birçok yazılım güvenilirlik modeli bu modelin referans alınmasıyla geliştirilmiştir [24]. Jelinski Moranda modelinin bazı varsayımları vardır. Bunlar [25];

- Yazılım başlangıçta N hata içerir.
- Her hata birbirinden bağımsızdır ve test sırasında her hataya ulaşmanın olasılığı eşittir.

- Hata oluşumları arasındaki süreler birbirinden bağımsızdır.
- Bir hata oluştuğunda bu hata kesinlikle giderilir.
- Bir hata oluştuğunda bu hata hemen giderilir ve hata düzeltme işlemi yeni bir hataya neden olmaz.
- İki hata oluşumunun arasında yazılımdaki bozulma oranı sabittir ve yazılımda bulunan hata sayısı ile orantılıdır.

Jelinski ve Moranda'nın bu varsayımlarla oluşturduğu modelin $i.$hata aralığındaki yazılım bozulma oranı;

$$\lambda(t_i) = \phi[N - (i-1)]$$

eşitliği ile hesaplanır. Eşitlikte ϕ model katsayısını, N başlangıçta yazılımda bulunan hata sayısını, t_i ise $(i-1).$ve $i.$ hataların arasındaki süreyi belirtir [22].

2.5.2.2.2. Schick Wolverton Modeli

Schick Wolverton modeli Jelinski Moranda modelinin değiştirilmesi ile elde edilir. Schick Wolverton modelinin Jelinski Moranda modelinden tek farkı $i.$ zaman aralığında bozulma oranı $t_i$test fazı arttıkça artar varsayımıdır [7, 24].

Modelde $i.$ve $(i-1).$ zaman aralığındaki bozulma oranı fonksiyonu aşağıdaki gibidir;

$$\lambda(ti) = \phi(N - (i-1))ti$$

Fonksiyondaki (ϕ , N), parametreleri Jelinski Moranda modeli ile aynıdır [22].

2.5.2.2.3. Jelinski Moranda Geometrik Modeli

Jelinski Moranda Geometrik modeli programda başlangıçta bozulma oranını sabit D ve zamanla geometrik olarak azaldığını varsayar. $i.$ hata zaman aralığındaki bozulma oranı ve güvenilirlik fonksiyonu aşağıdaki gibidir [7, 22]:

$$\lambda(t_i) = D * \left(k^{i-1}\right)$$

$$R(t_i) = e^{-Dk^{i-1}T_i}$$

$D =$ yazılımdaki başlangıç bozulma oranı

$K =$geometrik fonksiyon parametresi (0<k<1)

2.5.2.2.4. Moranda Geometrik Poisson Modeli

Moranda Geometrik Poisson dağılımı zaman aralıklarını T $2T$ eşit uzunlukta olacak şekilde sabitlemiş ve $i.$ zaman aralığında oluşan hata sayısının N_i , Dk^{i-1} ortalama ile poisson dağılım göstereceğini varsaymıştır [22].

$i.$ zaman aralığında m adet hata oranı;

$$P(N_i = m) = \frac{(e^{-Dk^{i-1}}(Dk^{i-1})^m)}{m!}$$

şeklindedir.

2.5.2.2.5. Değiştirilmiş Schick Wolverton Modeli

Sukert, Schick Wolverton modelini değiştirerek herhangi bir zaman aralığında birden çok hata oluşmasını varsaymıştır. Yazılım güvenilirlik oranı,

$$\lambda(t_i) = \phi(N - n_{i-1})t_i$$

şeklinde tanımlanır.

$(i-1).$ zaman aralığında tespit edilen toplam hata sayısı n_{i-1}'dir ve güvenilirlik fonksiyonu;

$$R(t_i) = e^{-\phi(N-n_{i-1})\frac{t_i^2}{2}}$$

şeklindedir [22].

2.5.2.2.6. Hata Düzeltme Sürecinin Tamamlanmadığı Durumlarda Goel Okumoto Modeli

Goel Okumoto modeli Jelinski Moranda modelini geliştirerek oluşan hataların giderilmesinin p olasılıkla gerçekleşebileceğini varsaymaktadır. Düzeltme sürecinin tamamlanmadığı Jelinski Moranda modelinde $i.$ hata zaman aralığında hata oranı fonksiyonu aşağıdaki gibidir [7, 22];

$$\lambda(t_i) = \phi(N - p(i-1))$$

Güvenilirlik fonksiyonu;

$$R(t_i) = e^{-\phi(N-p(i-1))t_i}$$

şeklindedir.

2.5.2.3. Güvenilirlik Geliştirme Modelleri

Güvenilirlik geliştirme modelleri test süreci boyunca yazılım güvenilirliğinin gelişimini ölçer ve tahmin eder. Güvenilirlik geliştirme modeli güvenilirliğin ya da bozulma oranını, zamanın ya da test durumu sayısının bir fonksiyonu olarak ifade eder. En çok bilinen yazılım güvenilirlik geliştirme modellerinden biri Wall Ferguson Modeli'dir.

2.5.2.3.1. Wall Ferguson Modeli

Wall Ferguson, Weibul geliştirme modeline benzer bir model ortaya sunmuşlardır. Modele göre, t anına kadar bulunan hataların kümülatif toplamları;

$$m(t) = a_0[b(t)]^{\beta}$$

denklemi ile hesaplanır. Eşitlikte $m(t)$, t anındaki kümülatif hata toplamını belirtirken, a_0 ve β bilinmeyen katsayıları ifade eder. Eşitlikteki $b(t)$, t anına kadar gerçekleşen test durumlarını veya test zamanını gösterir [22].

2.5.2.4. Homojen Olmayan Poisson Süreci Modelleri

Homojen Olmayan Poisson Süreci modelleri test sırasında ortaya çıkan hata durumlarının davranışını tanımlamak için analitik bir çözüm sunar [26]. Bu tür modellerin analitik çözüm sunması yazılım sektöründe sıklıkla kullanılmalarını sağlamıştır. Homojen Olmayan Poisson Süreci modellerinin temel özelliği belirli bir zamana kadar gözlemlenmesi muhtemel hataların kümülatif toplamının ortalama değer fonksiyonunu tahmin etmeleridir [20].

Homojen Olmayan Poisson Süreci modellerinde $m(t)$ t anına kadar gözlemlenmesi muhtemel hataların kümülatif toplamının ortalama değer fonksiyonudur. Genel olarak tüm Homojen Olmayan Poisson Süreci modelleri için ortalama değer fonksiyonu;

$$m(t) = \int_0^t \lambda(s)ds$$

eşitliği ile hesaplanır. Eşitlikte $\lambda(t)$ hata yoğunluk fonksiyonudur. Güvenilirlik ise;

$$R(t) = e^{-m(t)}$$

eşitliği ile ifade edilir.

Homojen Olmayan Poisson Süreci model türündeki bir modelin özelliklerini aşağıdaki fonksiyonlar açıklar:

$a(t)$: hata miktarı fonksiyonudur, t anında yazılımda tespit edilen ve henüz tespit edilmemiş hataların toplamıdır [27]. Bu fonksiyon bazı modellerde sabit değer alırken, bazı modellerde zamana göre artan bir değere sahip olabilir. Hata miktarı fonksiyonunun sabit olması yazılıma yeni hata eklenmesinin yok sayıldığı anlamına gelir. Artan değere sahip olduğunda zamanla yazılıma yeni hata eklenebiliyor demektir ve test ve hata düzeltme faaliyetlerinde yazılıma yeni hata eklenebilir manasına gelir.

$b(t)$: t anında hata tespit etme oranıdır [27].

Homojen olmayan yazılım güvenilirlik modelleri aşağıdaki varsayımlara dayanır [22, 28]:

- Yazılımda hataların oluşması rassaldır ve birbirinden bağımsızdır
- t anına kadar oluşmuş hataların kümülatif toplamı homojen olmayan poisson sürecini takip eder.
- Herhangi bir zamanda yazılım hata yoğunluğu oranı yazılımda henüz tespit edilmemiş hata sayısı ile orantılıdır.
- Yazılımda bir hata tespit edildiğinde hatayı düzeltmek için faaliyetler gerçekleştirilir ve bu hata düzeltme faaliyetleri her bir hatanın konumundan bağımsızdır.

En çok bilinen Homojen Olmayan Poissson Süreci yazılım güvenilirlik modelleri aşağıdaki gibidir.

- Üstel Homojen Olmayan Poisson Süreci Goel Okumoto Modeli
- Musa Üstel Modeli
- Geçiktirilmiş S Şekilli Homojen Olmayan Poisson Süreci Modeli
- Büklümlü S Şekilli Homojen Olmayan Poisson Süreci Modeli
- Yamada Homojen Olmayan Poisson Süreci Modeli
- Pham Nordmann Zang – Kesin olmaya Hata Tespiti Modeli (PNZ-IFD) [22]

2.5.2.4.1. Üstel Homojen Olmayan Poisson Süreci Goel Okumoto Modeli

Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modelinde, yazılım içerisinde zaten bulundurduğu hatalar nedeniyle rasgele zamanlarda bozulma yaşadığını varsaymıştır [24] .

Goel Okumoto modeli aşağıdaki varsayımlara dayanır [29]:

- Programdaki tüm hatalar hata tespit etme yönteminden bağımsızdır.
- Tespit edilen hatalar, bir sonraki testten önce kaldırılır.
- Her bir tespit edilen hata kaldırılır ve bu hatanın kaldırılması sırasında yeni hata oluşmasına sebep olunmaz
- Hataların meydana gelme ve tespit edilme olasılığı eşittir

Modelin diferansiyel denklemi [30]:

$$\frac{\partial m(t)}{\partial t} = b\left[a - m(t)\right]$$

denklemdeki a yazılımda tespit edilen ve henüz tespit edilmeyen hata sayısını gösterirken b hata tespit oranını göstermektedir. Diferansiyel denklemin çözümünden elde edilen ortalama değer fonksiyonu aşağıdaki gibidir:

$$m(t) = a(1 - e^{-bt})$$

2.5.2.4.2. Musa Üstel Modeli

Çalışma zamanı ve takvim zamanı arasındaki ilişki düşünülerek Goel Okumoto modeline benzer bir model ortaya çıkarılmıştır. Modele göre t anına kadar gözlemlenen hataların kümülatif toplamının ortalama değer fonksiyonu $m(t)$ ile diferansiyel denklem oluşturulmuştur.

$$\frac{\partial m(t)}{\partial t} = \frac{c}{nT}\left[a - m(t)\right]$$

Yukarıdaki diferansiyel denklemde a sistemde başlangıçta bulunan hata sayısını ifade eder, c ise n ve T ile beraber yazılımdaki bozulma oranını belirler.

T test başlangıcında hatalar arasındaki sürenin ortalama değeri, n ise yazılımda bulunması muhtemel hataların toplamını belirtir. Ayrıca t çalışma zamanı ya da işlemci zamanıdır yani test koşulma zamanını belirtir.

Bu denklemin çözümü ile Musa Üstel Homojen Olmayan Poisson Süreci yazılım güvenilirlik modeli aşağıdaki gibidir [22]:

$$m(t) = a(1 - e^{-ct/nT})$$

2.5.2.4.3. Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci Modeli

Yamada tarafından geliştirilmiş olan geciktirilmiş S şekilli modele göre yazılımda hata tespit etme süreci S şekilli bir grafikle tanımlanır. Modele göre test ekibinin test edecekleri yazılımı tanıması zaman alır. Bu nedenle ilk etapta hata tespit oranı düşüktür. Test ekibi yazılımı tanımaya başladıkça daha çok durumu test eder ve yazılım tespit oranı artmaya başlar. Her modelde olduğu gibi zamanla yazılımda bulunan hata sayısı azalacağından hata tespit oranı tekrar düşmeye başlar. Bu süreci dikkate alarak geliştirilen ve Homojen Olmayan Poisson Süreci özellikleri taşıyan modelin varsayımları aşağıdaki gibidir [22]:

- Her hata birbirinden bağımsızdır ve her hata için tespit edilme olasılığı eşittir.
- Herhangi bir anda yazılımda hata gözlemleme oranı, o anda yazılımda bulunan hata sayısı ile orantılıdır.
- Hata tespit etme olasılığı zamandan bağımsızdır.
- Başlangıçta yazılımda bulunan hata sayısı rasgele değişkendir.
- Yazılımda bulunan bir hata rasgele bir zamanda ortaya çıkabilir.
- $(i-1)$. ve i. hatalar arasındaki süre $(i-1)$. hatanın zamanına bağlıdır.
- Bir hata tespit edildiğinde bu hata hemen düzeltilir ve düzeltme işlemi yeni bir hataya neden olmaz.

Yamada'nın bu varsayımlarla geliştirdiği geciktirilmiş S şekilli yazılım güvenilirlik modeline göre yazılımda hata tespitinin zamana bağlı fonksiyonu $b(t)$;

$$b(t) = \frac{b^2 t}{1 + bt}$$

eşitliğindeki gibidir. Burada b herhangi bir hata için hata tespit oranını belirler ve verilen $b(t)$ fonksiyonu baz alınarak geliştirilen modelin ortalama değer fonksiyonu;

$$m(t) = a(1 - (1 + bt)e^{-bt})$$

şeklindedir.

Bu modeli kullanmanın avantajı, modelin hata izolasyonu veri analizinde kullanılabilmesidir. Hata izolasyonu, bir hatanın tanımlanması ve giderilmesi için bozulmanın tekrar yaratılması işlemidir [31].

2.5.2.4.4. Büklümlü S Şekilli Homojen Olmayan Poisson Süreci Modeli

Homojen Olmayan Poisson Süreci yazılım güvenilirlik modellerinin bir türü de S şekilli modellerdir. Bu tür modellerin ortalama değer fonksiyonu, üstel fonksiyonun bir defa kırılıp S şeklini almış halidir. Bu modeller şekillerinden dolayı S şekilli model olarak isimlendirilir.

Obha tarafından geliştirilen S şekilli yazılım güvenilirlik modeli büklümlü S şekilli yazılım güvenilirlik modeli olarak bilinir.

Obha'nın bu modeli geliştirirkenki varsayımları aşağıdaki gibidir [22]:

- Yazılımda bulunan hatalar tespit edilebilir ve tespit edilemez diye ikiye ayrılır ve bazı hatalar başka hatalar daha önceden bulunmadan bulunamaz hatalardır.
- Hata tespit oranı yazılımda o anda bulunan tespit edilebilir hataların sayısı ile orantılıdır.
- Tespit edilebilir her hatanın oluşma olasılığı sabit ve eşittir.
- Tespit edilemeyen hatalar da tamamen çözülebilir.

Obha'nın bu varsayımları kullanarak geliştirdiği yazılım güvenilirlik modelinin ortalama değer fonksiyonu aşağıdaki gibidir [31]:

$$m(t) = \frac{a(1 - e^{-bt})}{(1 + \beta e^{-bt})}$$

Eşitlikteki β parametresi büklüm derecesidir.

Büklümlü S şekilli modelin ortalama değer fonksiyonundan da anlaşılacağı üzere, yazılımda başlangıçta a tane hata vardır ve bu a tane hata zamanla değiştirilemez Yazılımda hata tespit edilmesi ise $b(t)$ fonksiyonu ile tanımlanmıştır.

Büklümlü S Şekilli Homojen Olmayan Poisson Süreci modeli, yazılım güvenilirlik modelleri içerisinde esnek bir model olarak bilinir. Parametre değerlerine göre hem üstel hem de S şekilli eğrilere uyum sağlayabilir ve yazılım hata verilerine uyum sağlamada kullanışlı bir modeldir [32].

2.5.2.4.5. Yamada Modeli

Yamada, Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci Modelini geliştirdiğinde yazılımda tespit edilen hatanın düzeltilmesi sırasında yazılıma yeni hata eklenmeyeceğini varsaymıştı. Yine Yamada hata düzeltme sırasında yapılan işlemle yazılıma yeni hatalar eklenebileceğini ve bunun da modelde belirtilmesi gerektiğini düşünüp kendi ismi ile anılan Yamada modelini geliştirmiştir.

Homojen Olmayan Poisson Süreci özelliği taşıyan Yamada modeline göre;

- Bir hata düzeltildiğinde bu yeni hataların oluşmasına neden olabilir.
- Yazılımda bir hata bulmanın olasılığı yazılımda kalan hataların sayısı ile orantılıdır.

Bu varsayımlarla Yamada aşağıdaki hata içerik fonksiyonunu ileri sürmüştür:

$$a(t) = ae^{\alpha t}$$

Bu hata içerik fonksiyonu ile;

$$m(t) = be^{-bt}\int_0^t a(s)e^{bs}ds$$

eşitliğini kullanarak aşağıdaki ortalama değer fonksiyonunu elde etmiştir.

$$m(t) = \frac{ab(e^{\alpha t} - e^{-bt})}{(\alpha + b)}$$

Bu model aynı zamanda Yamada hata düzeltme süreci tamamlanmamış model olarak da bilinmektedir. Bu modelde yazılımda tespit edilen ve henüz tespit edilmemiş tüm hataların sayısının α parametresine göre arttığı anlaşılmaktadır. Modelde hata tespit oranı sabittir [22].

2.5.2.4.6. Pham Nordmann Zang – Kesin Olmayan Hata Tespiti Modeli (PNZ-IFD)

Şimdiye kadar geliştirilen güvenilirlik modellerinde yeni bir hatanın test sırasında eklenebileceğini varsayan modeller açıklanmıştır ve hata içerik fonksiyonu test zamanına bağımlı bir fonksiyondur. Bu durum hata içerik oranı hakkında bazı varsayımlar yapılmasını gerektirmektedir. Pham, Nordmann ve Zang [22] hata tespitini hata üretme ve test kapsamına bağlayan yeni bir model geliştirmişlerdir.

Fakat bu model hata içerik fonksiyonu hakkında herhangi bir varsayım içermemektedir ve bu sebeple yazılımdaki tespit edilen ve henüz tespit edilmemiş hata sayısı zamanla değişiklik göstermez [22].

Pham, Nordmann ve Zhang aşağıdaki varsayımları Bölüm 2.5.2.4'te verilen genel varsayımlara ekleyerek PNZ IFD modelini geliştirmiştir:

- Tespit edilen hatayı düzeltme faaliyetleri sırasında yazılıma yeni hatalar eklenebilir ve bu hataların eklenmesinin yoğunluğu $d(t)$ fonksiyonu ile belirtilir.
- Hataların eklenme oranının test faaliyetleri ile azaldığı varsayılır ve test faaliyetlerinin sonuna doğru hata eklenme oranı önemsiz bir değer alır.

Bu varsayımlarla PNZ IFD modelinin denklemi aşağıdaki gibidir:

$$\frac{dm(t)}{dt} = \frac{c'(t)}{1-c(t)}\left[a-m(t)\right] - d(t)\left[a-m(t)\right]$$

Denklemde $c(t)$ test kapsamı fonksiyonudur ve t anına kadar test durumlarının kapsamış olduğu yazılım kodunun yüzdesini ifade eder.

Pham, Nordmann ve Zang'ın 2004 yılında geliştirmiş oldukları test kapsamı fonksiyonu;

$$c(t) = 1-(1+b(t))e^{-bt}$$

ve hataların eklenmesi yoğunluk oranı ise;

$$d(t) = \frac{d}{1+dt}$$

denklemlerindeki gibidir.

Verilen $c(t)$ ve $d(t)$ denklemlerini PNZ denkleminde yerine konulduğunda ortalama değer fonksiyonu aşağıdaki gibidir:

$$m(t) = a - ae^{-bt}\left[1+(b+d)t+bdt^2\right]$$

Bu model aynı zamanda PNZ-Kesin Olmayan Hata Tespiti yazılım güvenilirlik modeli olarak da bilinir [22].

2.6. Parametrelerin Tahmin Edilmesinde Kullanılan Yöntem

Güvenilirlik modellerinde bulunan parametrelerin tahmin edilmesinde kullanılmak üzere tahmin yöntemleri vardır. Bu yöntemler, En Çok Olabilirlik Tahmin Yöntemi ve Doğrusal Olmayan En Küçük Kareler Yöntemidir. Güvenilirlik modelleri, En Çok Olabilirlik Tahmin Yöntemi ve Doğrusal Olmayan En Küçük Kareler Yöntemini kullanılarak hata kayıt verileri ile uyumlandırılabilir. Zhao ve Xie'nin [33] belirttiği üzere Homojen Olmayan Poisson Süreci yazılım güvenilirlik modellerinin parametre tahmininde en sık kullanılan yöntem olmasından dolayı bu çalışma kapsamında yazılım güvenilirlik modellerinin parametre tahmininde En Çok Olabilirlik Tahmin yöntemi kullanılmıştır [12, 19].

2.6.1. En Çok Olabilirlik Tahmin Yöntemi

En çok olabilirlik tahmin yöntemi, matematiksel bir modelin istenilen bir veri setine eşleştirilmesi için kullanılan popüler ve kullanışlı bir istatistiksel yöntemdir. Yöntemin temel aldığı fikir örnek veri setinin olasılığını maksimize eden parametreleri tahmin etmektir [13]. Farklı modellerde ve farklı tür veri setleri için kullanılabilir olması yöntemi popüler yapmıştır. En çok olabilirlik yönteminin uygulanması basit görünse de matematiksel olarak yoğun işlem gücü gerektiren bir yöntemdir fakat günümüz bilgisayarları ile yöntemin uygulanması kolaylaştırılmaktadır.

$f(x;\theta)$ dağılımının rassal değişkenleri $x_1, x_2, ..., x_n$ olsun

En çok olabilirlik tahmin fonksiyonu aşağıdaki gibidir:

$$L(\theta; x_1, x_{2,}..., x_n) = f(x_1, \theta) f(x_2, \theta)... f(x_n, \theta)$$

En çok olabilirlik tahmin edicisi θ'nın bir fonksiyonu olan L olabilirlik fonksiyonunu maksimum yapan θ parametresinin $\hat{\theta}$ değerine en çok olabilirlik tahmin edicisi denir.

En çok olabilirlik yöntemi kullanılarak model parametrelerinin tahminlenmesi için aşağıdaki adımlar uygulanmalıdır:

- Güvenilirlik modellerinde parametrelerin tahmin edilebilmesi için ilk olarak modele ait en çok olabilirlik tahmin fonksiyonu, yoğunluk fonksiyonundan bulunmalıdır.
- En çok olabilirlik fonksiyonunun doğal logaritması alınır.

- Tahmin edilecek her bir parametre için doğal logaritması alınan en çok olabilirlik fonksiyonunun kısmi türevi alınır.
- Türev sonucu elde edilen denklem sıfıra eşitlenir.
- Son olarak elde edilen eşitlikler çözülerek parametreler tahmin edilir.

Son adımdaki eşitliklerin analitik bir çözümü olmayabilir, bu nedenle numerik çözümle parametreler bulunabilir [34].

2.7. İlişkili Çalışmalar

Bilgisayar yazılımları günlük hayatın vazgeçilmez bir parçası haline gelmiştir ve birçok sektörde yazılımların güvenilirliğine olan ihtiyaç gün geçtikçe artmaktadır [12]. Yazılım güvenilirliğine olan ihtiyacın artmasıyla, yazılım güvenilirliğinin ölçülmesi ve tahmin edilebilmesine olan ihtiyaç da artmıştır. Bu nedenle son yıllarda yazılım güvenilirliğine yönelik çalışmalar önem kazanmaktadır. Yapılan çalışmalar sonucu yazılım güvenilirliğinin tahmin edilmesi amacıyla kullanılabilecek birçok yazılım güvenilirlik modeli geliştirilmiştir [22]. Bununla birlikte hangi modelin ne zaman hangi durumlarda kullanılabileceği yeni bir sorun oluşturmuştur ve son yıllarda farklı durumlarda farklı yazılım güvenilirlik modellerinin performanslarının karşılaştırılması ile ilgili çalışmalar yoğunluk kazanmıştır. Örneğin; Ullah, Morisio ve Vetro'nun 2012 tarihli çalışmasında [35], sekiz yazılım güvenilirlik (Musa Okumoto, Büklümlü S Şekilli, Goel Okumoto, Geciktirilmiş S Şekilli, Lojistik, Gompertz, Yamada Üstel ve Genelleştirilmiş Goel Model) modeli 50 farklı hata kayıt veri seti için ayrı ayrı performansları açısından test edilmiş ve hangi modellerin hangi durumlarda hata oluşumunu daha iyi tahmin ettiği araştırılmıştır. Araştırma sonucunda Musa Okumoto modelinin endüstriyel projelerdeki veri setlerini modellemede ve hata tahmininde en iyi sonucu verdiği görülmüştür. Açık kaynak yazılımlarında ise Gompertz modelinin ve Büklümlü S Şekilli modelinin iyi sonuçlar verdiği belirtilmektedir. Aydın'ın 2014 tarihli çalışmasında ise [9] hata yoğunluğunu tahminlemenin proje geliştirme yaşam döngüsü ile ilişkili olabileceği düşünülmüş ve iteratif yaşam döngüsüne sahip bir proje ve projenin modülleri için Rayleigh modeli ve doğrusal regresyon modelinin performansları karşılaştırılmıştır. Karşılaştırma sonucunda modül seviyesinde Rayleigh modeli proje seviyesinde ise Doğrusal Regresyon modelinin daha iyi sonuç verdiği gözlemlenmiştir.

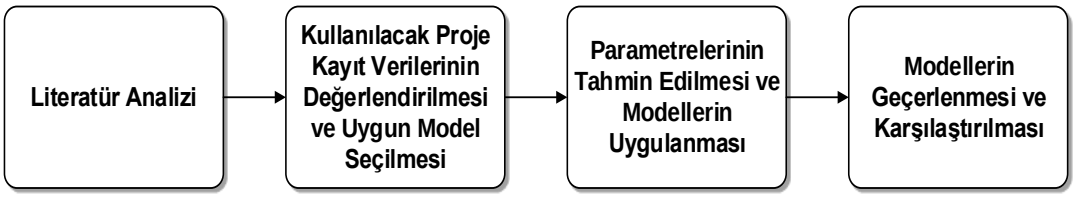
Homojen Olmayan Poisson Süreci modelleri yazılım projelerinde güvenilirlik tahmininde kullanımı gittikçe artan yazılım güvenilirlik modelleri arasındadır [20, 22]. Bu nedenle literatürde bulunan ve Homojen Olmayan Poisson Süreci modellerinin performanslarının karşılaştırılmasını amaçlayan çalışmalar yapılmıştır. Pham [12] bir çalışmasında Homojen Olmayan Poisson Süreci modellerinin performanslarını karşılaştırırken, çevresel faktörler ve bütçe modelinin yazılım güvenilirliğine etkilerini araştırmıştır. Yazılım güvenilirlik modellerinin karşılaştırması yapılır iken belirli performans ölçümleri kullanılmaktadır. Hangi tür performans ölçümlerinin kullanılması da ayrı bir araştırma konusudur. Downs ve Scott [36], Littlewood tarafından önerilen ölçüm yönteminin yazılım güvenilirlik modellerinin performanslarının ölçülmesinde kullanılıp kullanılamayacağı incelenmiş ve Littlewood tarafından önerilen yöntemin geliştirilmesi yapılmıştır.

Bir modelin performansı o modelin proje hata kayıt verilerine ne kadar uyum sağlayabildiğiyle ilgilidir. Her yazılım güvenilirlik modeli istenilen hata kayıt verilerine uyum sağlayabilmesi için ayarlanabilen parametrelere sahiptir. Bu parametrelerin nasıl ve neye göre belirleneceği ile ilgili yapılan bir çalışmada [7] bir modelin istenilen hata kayıt verilerine uyarlanabilmesi için adım adım izlenmesi gereken bir prosedür önerilmiştir. Ayrıca model parametrelerinin nasıl tahmin edileceği ile ilgili başka bir çalışmada [19] otomotiv sektöründeki yazılım projelerinde yazılım güvenilirlik modellerinin parametrelerinin tahmininde yöntemler araştırılmıştır ve araştırma sonucunda V Modelin, Agile yazılım yaşam döngülerine sahip projelerde Gompertz modelinin, Şelale yazılım yaşam döngüsüne sahip projelerde ise Lojistik modelin en iyi sonucu verdiği belirtilmiştir.

Bu çalışma kapsamında ise web yazılımlarında Homojen Olmayan Poisson Süreci güvenilirlik modellerinin hata kayıt verilerini modellemesindeki performansları karşılaştırılmıştır. Yapılan çalışmada bir yazılım güvenilirlik modeli için proje hata kayıt verilerine uygun parametrelerin nasıl belirlenebileceği, parametreleri bulunmuş modelin performans ölçümlerinin nasıl yapılabileceği ve farklı modellerin performanslarının nasıl karşılaştırılabileceği açıkça anlatılmıştır.

3. METODOLOJİ

Bu çalışma kapsamında, hata bilgilerinin düzenli olarak kaydedildiği dört adet web yazılımı projesine ait yazılım güvenilirlik modellerinin karşılaştırılması gerçekleştirilmiştir. Tez çalışması kapsamında Şekil 3 sunulan fazlar planlanmış ve plan doğrultusunda çalışma gerçekleştirilmiştir.



Şekil 3 Bu Çalışmada İzlenen Adımlar

3.1. Kullanılan Proje Kayıt Verileri

Tez kapsamında yazılım güvenilirlik modellerini karşılaştırmak için ALTAIR Yazılım ve Savunma Teknolojileri A.Ş firması tarafından geliştirilen projelerin olay kayıtları kullanılmıştır. ALTAIR Yazılım ve Savunma Teknolojileri A.Ş. Ocak 2014 tarihinde kurulmuş, ISO 9001: 2008 Kalite Sertifikasına sahip, 8 adet yazılım ürünü geliştirmiş ve yeni yazılım ürünleri geliştirmeye devam etmekte olan bir yazılım firmasıdır. Firmada 08.11.2015 tarihi itibariyle 30'a yakın kişi çalışmaktadır ve bu kişiler en az bir proje ile ilgilenen ekiplere ayrılmaktadır. Şirkette çalışan kişi sayısı ve kişiler çok fazla değişikliğe uğramamıştır. Proje çalışanları, geliştirme ekibi, testçi, proje yöneticisi ve kalite güvence uzmanından oluşur.

Geliştirme ekibi iç ya da dış müşteriden gelen gereksinimler doğrultusunda gereksinimlerin oluşturulması, yazılımın geliştirilmesi, yazılımın bakımının yapılması, birim testlerinin gerçekleştirmesi, testler sırasında tespit edilen hataların hata yönetim aracına kaydedilmesi ve hataların giderilmesinden sorumludur. Projedeki testçiler yazılımın sistem testlerinin gerçekleştirilmesinden ve testler sırasında tespit edilen hataların hata yönetim aracına kaydedilip, geliştirme ekibine aktarılmasından ve hatalar geliştirme ekibi tarafından giderildikten sonra bu hataların giderilmesinin kontrol edilmesinden sorumludur. Kalite güvence uzmanı, kalite yönetim sisteminin yeterlilik, amaca uygunluk ve etkinlik yönünden sistematik ve periyodik olarak iç kalite denetimlerini yapar. Kalite güvence uzmanı projelere ait

hata kayıt verileri gibi ölçümleri hata yönetim aracından/proje yönetim aracından toplayarak analiz edilmesi ve ölçüm sonuçlarının proje paydaşları ile paylaşılması, gerekli düzeltici faaliyetlerin başlatılması ve izlenmesinden sorumludur. Proje yöneticisi ise yazılım geliştirme projesi kapsamında yürütülecek olan etkinlikleri, iş adımlarını ve kaynakları belirlemek, planlamak, izlemek, projenin paydaşları ile olan koordinasyonu sağlamak ve gerekli görüldüğünde düzeltici önlemler almaktan sorumludur.

Tez kapsamında firmanın dört ayrı projesi için geliştirme süreci inceleme altına alınmıştır. Yazılımların geliştirme sürecinde birim testleri geliştirme ekibi tarafından yapılıp hata kayıtları JIRA kullanılarak kaydedilmiştir. Projelerin plan ve takvimlerine göre sistem testleri test ekibi tarafından gerçekleştirilmiş ve hata kayıtları yine JIRA kullanılarak kayıt altına alınmıştır. Testlerin yapılıp gerekli tüm kayıtların saklanması kalite güvence uzmanı tarafından denetlenmiştir. Bu faaliyetlerin gerçekleştirilmesi sonucu dört projede de hata kayıtlarının düzenli bir şekilde saklanması sağlanmıştır. Tez çalışması sırasında hata kayıt verilerine istenilen herhangi bir zamanda ulaşılmıştır. Çalışma süresince devam eden projeler için yeni kayıtlar tekrar tekrar alınmıştır ve bu çalışma 08.11.2015 tarihine kadar olan verilere dayanmaktadır.

Firma proje isimlerinin kullanılmasına yetki vermediğinden projeler “Proje A”, “Proje B”, “Proje C”, “Proje D” olarak isimlendirilmiştir. Üzerinde çalışılan projelerin kısa özeti Çizelge 1 Proje Özeti‘de verilmiştir.

Çizelge 1 Proje Özeti

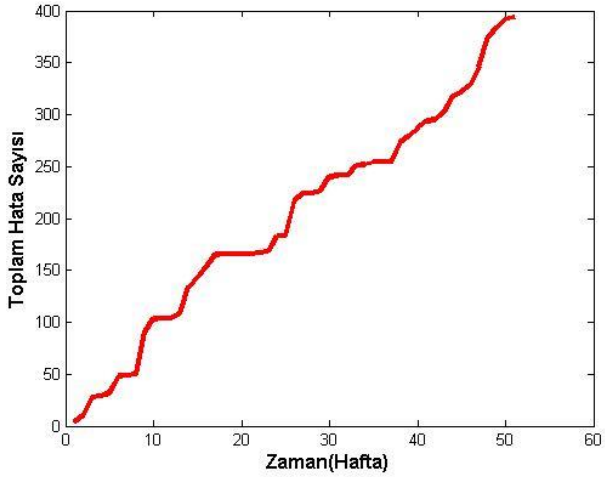
Gerekli Bilgi	**Proje A**	**Proje B**	**Proje C**	**Proje D**
Personel Sayısı	4	3	3	2
Yaşam Döngüsü	Spiral	Spiral	Şelale	Şelale
Kod Satır Sayısı[1]	40000	10300	12900	13343
Devam Etme Durumu	Devam	Devam	Bitti	Bitti
Müşteri	AR-GE	AR-GE	Müşteri X	Müşteri Y
Testçi Sayısı	1	1	1	1
Yazılım Geliştirme Ortamı	Windows Eclipse	Windows Eclipse	Windows Eclipse	Windows Eclipse
Yazılım Dili	Java	Java	Java	Java
Yazılımın Çalışacağı Platform	Linux	Linux	Linux	Linux
Uygulanan Test Tipi	Birim, Sistem	Birim, Sistem	Birim, Sistem	Birim, Sistem

[1] Kod satır sayısı devam eden projeler için 08.11.2015 tarihine kadar olan kod satır sayısıdır.

ALTAIR Yazılım ve Savunma Teknolojileri A.Ş. tarafından alınan kayıtlarda sadece hata kayıtları değil aynı zamanda yazılımın geliştirme sürecinde yazılımda meydana gelen değişikliklerle ilgili olay kayıtları da bulunmaktadır. Bu tez kapsamında, alınan kayıtlardan sadece hatalara ilişkin bilgiler kullanılmıştır. Hata kayıtlarında hataların tespit edilme zamanları alınmış ve her bir proje için birim zamanda görülen hata sayısı bilgisi elde edilmiştir. Her bir proje için birim zamanda görülen hata sayıları Çizelge 2 Proje A Hata Kayıtları - Çizelge 5 Proje D Hata Kayıtları ve Şekil 4 Proje A Hata Kayıtları - Şekil 7 Proje D Hata Kayıtları'nda verilmiştir.

Çizelge 2 Proje A Hata Kayıtları

Zaman (Hafta)	Hata Sayısı	Kümülatif Hata Sayısı	Zaman (Hafta)	Hata Sayısı	Kümülatif Hata Sayısı
1	4	4	**27**	7	224
2	6	10	**28**	0	224
3	18	28	**29**	3	227
4	1	29	**30**	13	240
5	3	32	**31**	1	241
6	16	48	**32**	0	241
7	1	49	**33**	10	251
8	1	50	**34**	1	252
9	41	91	**35**	2	254
10	12	103	**36**	0	254
11	1	104	**37**	0	254
12	0	104	**38**	19	273
13	5	109	**39**	6	279
14	25	134	**40**	8	287
15	9	143	**41**	7	294
16	11	154	**42**	1	295
17	11	165	**43**	8	303
18	1	166	**44**	15	318
19	0	166	**45**	4	322
20	0	166	**46**	7	329
21	0	166	**47**	17	346
22	1	167	**48**	29	375
23	1	168	**49**	9	384
24	15	183	**50**	8	392
25	0	183	**51**	2	394
26	34	217			

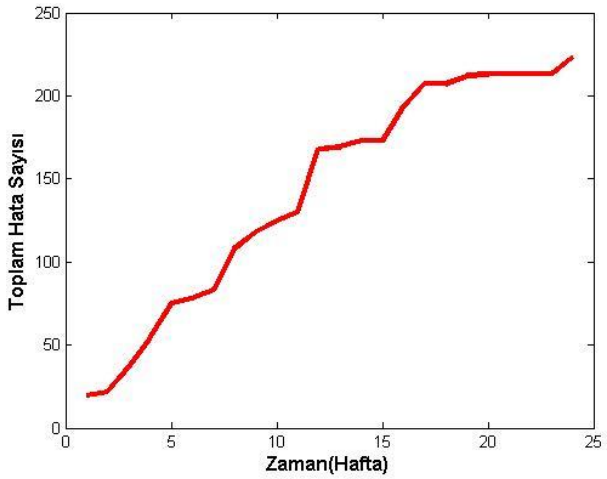


Şekil 4 Proje A Hata Kayıtları

Şekil 4 Proje A Hata Kayıtları incelendiğinde kümülatif hata kayıtlarının zamana göre yaklaşık doğrusal bir şekilde ilerlediği görülmektedir. Kümülatif hatanın doğrusal ilerlemesi projenin yaşam döngüsü modelinden kaynaklanabileceği gibi, projenin henüz olgunlaşmadığını ve başlangıç aşamasında olduğunu da gösterebilir. Bu projenin hata kayıt grafiği ilerleyen dönemde iç bükey ya da S şekline dönüşebilir.

Çizelge 3 Proje B Hata Kayıtları

Zaman (Hafta)	Hata Sayısı	Kümülatif Hata Sayısı	Zaman (Hafta)	Hata Sayısı	Kümülatif Hata Sayısı
1	20	20	**13**	1	169
2	2	22	**14**	4	173
3	15	37	**15**	0	173
4	17	54	**16**	20	193
5	21	75	**17**	14	207
6	3	78	**18**	0	207
7	5	83	**19**	5	212
8	25	108	**20**	1	213
9	10	118	**21**	0	213
10	7	125	**22**	0	213
11	5	130	**23**	0	213
12	38	168	**24**	10	223

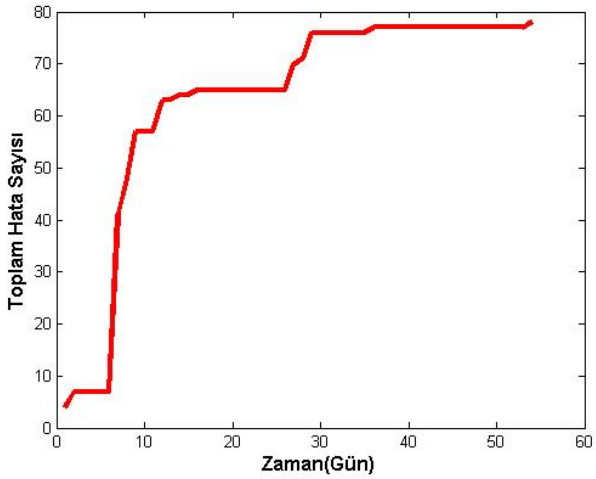


Şekil 5 Proje B Hata Kayıtları

Şekil 5 Proje B Hata Kayıtları incelendiğinde kümülatif hata kayıtlarının zamana göre iç bükey bir şekilde ilerlediği görülmektedir. Bu durum projede hata tespit hızının zamanla azaldığını ve bu nedenle kümülatif hatanın artışının gittikçe yavaşladığını gösterir.

Çizelge 4 Proje C Hata Kayıtları

Zaman (Gün)	Hata Sayısı	Kümülatif Hata Sayısı	Zaman (Gün)	Hata Sayısı	Kümülatif Hata Sayısı
1	4	4	**28**	1	71
2	3	7	**29**	5	76
3	0	7	**30**	0	76
4	0	7	**31**	0	76
5	0	7	**32**	0	76
6	0	7	**33**	0	76
7	34	41	**34**	0	76
8	7	48	**35**	0	76
9	9	57	**36**	1	77
10	0	57	**37**	0	77
11	0	57	**38**	0	77
12	6	63	**39**	0	77
13	0	63	**40**	0	77
14	1	64	**41**	0	77
15	0	64	**42**	0	77
16	1	65	**43**	0	77
17	0	65	**44**	0	77
18	0	65	**45**	0	77
19	0	65	**46**	0	77
20	0	65	**47**	0	77
21	0	65	**48**	0	77
22	0	65	**49**	0	77
23	0	65	**50**	0	77
24	0	65	**51**	0	77
25	0	65	**52**	0	77
26	0	65	**53**	0	77
27	5	70	**54**	1	78



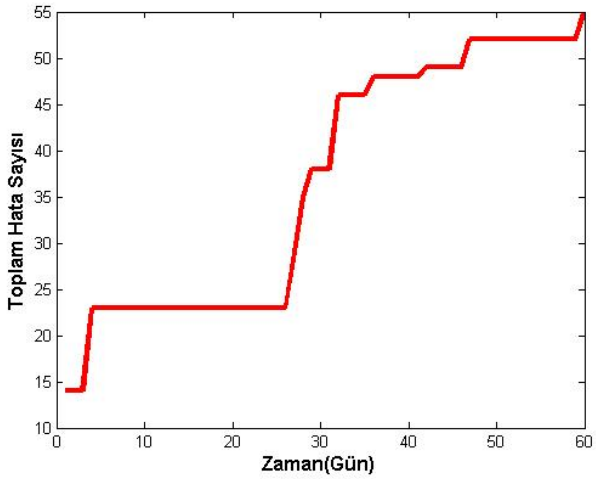
Şekil 6 Proje C Hata Kayıtları

Şekil 6 Proje C Hata Kayıtları incelendiğinde kümülatif hata kayıtlarının zamana göre iç bükey bir şekilde ilerlediği görülmektedir. Proje B'den farklı olarak bu projede yeni hata tespitinin neredeyse hiç olmadığı görülebilir. Zaten proje bilgilerinde de

projenin geliştirilmesinin tamamlandığı verilmiştir. Aynı durum hata kayıtlarından da anlaşılabilmektedir.

Çizelge 5 Proje D Hata Kayıtları

Zaman (Gün)	Hata Sayısı	Kümülatif Hata Sayısı	Zaman (Gün)	Hata Sayısı	Kümülatif Hata Sayısı
1	14	14	**31**	0	38
2	0	14	**32**	8	46
3	0	14	**33**	0	46
4	9	23	**34**	0	46
5	0	23	**35**	0	46
6	0	23	**36**	2	48
7	0	23	**37**	0	48
8	0	23	**38**	0	48
9	0	23	**39**	0	48
10	0	23	**40**	0	48
11	0	23	**41**	0	48
12	0	23	**42**	1	49
13	0	23	**43**	0	49
14	0	23	**44**	0	49
15	0	23	**45**	0	49
16	0	23	**46**	0	49
17	0	23	**47**	3	52
18	0	23	**48**	0	52
19	0	23	**49**	0	52
20	0	23	**50**	0	52
21	0	23	**51**	0	52
22	0	23	**52**	0	52
23	0	23	**53**	0	52
24	0	23	**54**	0	52
25	0	23	**55**	0	52
26	0	23	**56**	0	52
27	6	29	**57**	0	52
28	6	35	**58**	0	52
29	3	38	**59**	0	52
30	0	38	**60**	3	55



Şekil 7 Proje D Hata Kayıtları

Şekil 7 Proje D Hata Kayıtları incelendiğinde kümülatif hata kayıtlarının zamana göre S şekilli ilerlediği görülmektedir. Proje başlangıcında kümülatif hata artışı az iken ilerleyen zamanda gittikçe artmış ve sonra yine azalmıştır.

3.2. Kullanılan Araçlar

Yazılım güvenilirlik modellerinin ortalama değer fonksiyonlarındaki parametrelerin tahmin edilmesi için en çok olabilirlik yöntemi kullanılmıştır. En çok olabilirlik yönteminin uygulanması sonucu ortaya çıkan denklem kümelerinin çözümü için MATLAB[2] uygulaması kullanılmıştır. Denklemlerin tanımlanması için MATLAB üzerinde fonksiyonlar kodlanmıştır. Ayrıca MATLAB Optimization Toolbox[3] kullanılarak denklem çözümlenmesi yapılmıştır.

3.3. Proje Kayıt Verilerinin Değerlendirilmesi ve Uygun Modelin Seçilmesi

Yazılım Güvenilirlik Modelleri başlığında anlatıldığı üzere yazılım güvenilirliğinin hesaplanmasında kullanılan çok sayıda model bulunmaktadır. Fakat bu modellerin tümünün her durumda kullanılması mümkün değildir [3, 37]. Özellikle eldeki verilere uygun modellerin kullanılması gerekmektedir [19]. Literatürde bulunan yazılım güvenilirlik modelleri aynı veri üzerinde farklı tahmin sonuçları verebilir. Bu durum yalnızca yazılım güvenilirlik modelleri için değil tüm matematiksel modelleme türleri için geçerlidir. Bununla birlikte bir veri için uygun tahmin sonuçları veren bir model, diğer bir veri için uygun sonuç vermeyebilir. Bu durum yazılım güvenilirlik modeli kullanıcıları için önemli bir problem olmuştur. Yazılım güvenilirlik modeli kullanıcıları ilk olarak ne tür bir model kullanmaları gerektiğine karar vermelidirler [20]. Hangi tür model kullanılacağı her proje için proje şartlarına göre seçilmelidir. Bazı tür modellerde yazılıma müdahale gerekirken, bazı türlerde ise hata ortaya çıkma durumlarının anlık olarak kaydedilmesi gerekir.

Örneğin deterministik modeller yazılım güvenilirliğini yazılımın yapısal özelliklerini inceleyerek tahmin etmeye çalışmaktadır. Bu tür bir yazılım güvenilirlik modelinin kullanılması için yazılımın kodu, tasarım dokümanları, gereksinimleri ve zorluk dereceleri gibi verilere ihtiyaç duyulur. Deterministik yazılım güvenilirlik modelleri

[2] MATLAB R2013b, (The Mathworks, Inc., Natick, Massachusetts, United States)
[3] MATLAB Optimization Toolbox 6.3 (The Mathworks, Inc., Natick, Massachusetts, United States)

genellikle durağan analiz sonucu elde edilen veriler ile yapılan modellemelerdir. Diğer model türlerine nazaran bu modellerde hata kayıtları gerekmez.

Hata kaynağı modellerinde ise yazılımın kendisine eklemeler yapılması gerekmektedir. Yazılıma ekleme yapılması her durumda gerçekleşebilir değildir. Projenin bütçesinde ve takviminde artışa neden olabilir. Ayrıca yazılıma eklemeler yapılması yazılımın davranışını da değiştirebilir.

Güvenilirlik geliştirme modellerinde yazılımın güvenilirliği, zamanın veya test durumlarının bir fonksiyonu olarak ölçülür [22]. Bu tür modellerde geliştirilen test durumları ve test gerçekleşme zamanları modelin bir girdisidir. Bu nedenle güvenilirlik geliştirme modelinin kullanılabilmesi için test koşum zamanlarının bilinmesi gerekir.

Bu çalışma kapsamında hata kayıt verilerinden, belirli aralıklarda oluşan toplam hata sayıları elde edilmiştir. Elde edilen bu veri ile her bir projenin kümülatif toplam hata sayısı oluşturulmuş ve bu veri yazılım güvenilirlik analizinde kullanılmıştır. Homojen Olmayan Poisson Süreci modelleri de belirli aralıklarda gözlemlenen toplam hata sayısı verisini kullandıklarından, Homojen Olmayan Poisson Süreci modellerinin güvenilirlik analizinde kullanılması uygun görülmüştür [20]. Yazılım sektöründe Homojen Olmayan Poisson Süreci modelleri yazılım projelerinde güvenilirlik, kalan hata sayıları ve hata yoğunluğu gibi yazılım kalite özelliklerinin tahmin edilmesinde genellikle kullanılmaktadır [20, 22].

3.3.1. Yazılım Güvenilirlik Modellerinin Uygulanması

Homojen Olmayan Poisson Süreci modelleri yazılım güvenilirlik ölçümünde ortalama değer fonksiyonlarını kullanırlar. Çizelge 6 Kullanılan Yazılım Güvenilirlik Modelleri'nde bu çalışma kapsamında kullanılan modellerin ortalama değer, hata sayısı ve bozulma oranı fonksiyonları verilmiştir.

Çizelge 6 Kullanılan Yazılım Güvenilirlik Modelleri

Fonksiyon / Model Adı	Ortalama Değer Fonksiyonu (m_t)	Hata Sayısı Fonksiyonu (a_t)	Bozulma Oranı Fonksiyonu (b_t)
Üstel Homojen Olmayan Poisson Süreci Goel-Okumoto Modelleri	$m(t) = a(1 - e^{-bt})$	$a(t) = a$	$b(t) = b$
Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci Modelleri	$m(t) = a(1 - (1 + bt)e^{-bt})$	$a(t) = a$	$b(t) = \frac{(b^2 t)}{(1 + bt)}$
Büklümlü S Şekilleri Modelleri	$m(t) = \frac{a(1 - e^{-bt})}{(1 + \beta e^{-bt})}$	$a(t) = a$	$b(t) = \frac{b}{(1 + \beta e^{-bt})}$
Musa Üstel Modeli	$m(t) = a(1 - e^{-ct/nT})$	$a(t) = a$	$b(t) = \frac{c}{nT}$
Yamada Modeli	$ab(e^{\alpha t} - e^{-bt})$	$a(t) = ae^{\alpha t}$	$b(t) = b$
Pham-Nordmann-Zhang (PNZ) Kesin Olmayan Hata Tespiti Modeli	$m(t) = a - ae^{-bt}(1 + (b + d)t + bdt^2)$	$a(t) = a$	$b(t) = (b^{2t}/(1 + bt)) - (d/(1 + dt))$

Çizelge 6'da verilen ortalama değer fonksiyonları (m(t)), t anına kadar yazılımda gözlemlenen toplam hata sayısını modeller. Yazılımda bulunan hata sayısı fonksiyonu $(a(t))$ t anında yazılımında bulunduğu tahmin edilen hata sayısının fonksiyonudur. Bozulma oranı fonksiyonu $(b(t))$ t anında yazılımda bir hata gözlemleme olasılığını ifade eder.

ALTAIR Yazılım ve Savunma Teknolojileri A.Ş. firmasından elde edilen veriler ile belirli zamanlarda, projelerde gözlemlenen kümülatif hata verisi elde edilmiş ve bu veri, kullanılan modellerin ortalama değer fonksiyonları ile modellenmiştir.

3.3.2. Parametrelerin Tahmin Edilmesi

Kullanılan güvenilirlik modelleri, farklı projelere uygulanabilmesi için parametrelere sahiptir. Bu çalışma kapsamında uygulaması yapılan Homojen Olmayan Poisson Süreci modellerinin, ortalama değer fonksiyonlarında bulunan her bir parametre güvenilirlik modellemesi yapılacak proje için bilgiler verir. Güvenilirliğin doğru

modellenebilmesi için model parametrelerinin, proje ve eldeki veriye uygun olarak tahmin edilmesi gerekmektedir [31]. Model parametrelerinin doğru tahmin edilmesi model başarımı için oldukça önemlidir.

Bu çalışma kapsamında model parametrelerinin tahmin edilmesi için en çok olabilirlik tahmin yöntemi kullanılmıştır. İlk aşamada en çok olabilirlik tahmin yöntemi sayesinde en uygun parametre tahmini için denklem kümeleri üretilmiştir.

En çok olabilirlik tahmin yönteminin uygulanması ile elde edilen denklemler doğrusal değildir ve analitik çözümleri yoktur. Bu yüzden sayısal çözümler yapılmıştır. Bu amaçla MATLAB uygulaması kullanılmıştır. Her bir denklem seti için girilen hata kayıt verilerine göre denklemlerin sonucunu veren MATLAB fonksiyonları yazılmıştır. Üstel Homojen Olmayan Poisson Süreci Goel Okumoto, Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci, Büklümlü S Şekilleri ve Musa Üstel modellerinde denklemler daha basit olduğu için sayısal çözümlerde denemeler kodla yapılmış ve optimum sonuç alınmıştır. Yamada Modeli ve Pham-Nordmann-Zhang (PNZ) Kesin Olmayan Hata Tespiti modellerinde ise denklemlerin çözümü için MATLAB Optimization Toolbox 6.3 kullanılmış ve parametreler en uygun şekilde tahmin edilmiştir.

Bu çalışma kapsamında ele alınan dört proje ve kullanılan altı modelin her biri için parametre tahminleri yapılmıştır. Parametre tahminlerinde, eldeki proje hata kayıt verilerinin %100'ü, %70'i ve %50'si kullanılmıştır ve böylelikle her bir proje ve model için üç ayrı parametre tahmini yapılmıştır.

3.3.3. Model Performans Analizi

Güvenilirlik modelleri kurulduktan sonra, kurulan modellerin ne kadar iyi çalışıp gerçeğe yakın sonuçlar ürettiğini değerlendirebilmek için ilgili modellerin tahminleme performanslarının ölçülmesi gerekmektedir. Modellerin performansını ölçmekte kullanılan istatistiksel yöntemler, tahminleme sonucu elde edilen veriler ile gerçek verilerin karşılaştırmasını yaparak modelin uygunluğunu belirler [32]. Güvenilirlik modellerinin uyumlarının belirlenmesinde MSE, MMRE, PRED ve A.BPRE yöntemleri kullanılmıştır.

Bu çalışma kapsamında her bir proje ve model için üç ayrı parametre tahmini yapılmıştır. Yapılan her bir parametre tahmini yeni bir model oluşturmuştur ve her

bir modelin performansını ölçmek için MSE, PRED, MMRE, A.BPRE değerleri hesaplanmıştır.

Proje hata kayıt verilerinin %100'ünün parametre tahmini için kullanıldığı durumda, model performansını ölçmede kullanılabilecek veri kalmadığından, yine aynı veriler MSE, PRED, MMRE ve A.BPRE değerlerinin hesaplanmasında kullanılmıştır. Proje hata kayıt verilerinin %70 ve %50'sinin parametre tahmini amacıyla kullanıldığı durumda ise sırasıyla %30 ve %50'lik kalan veri MSE, PRED, MMRE ve A.BPRE değerlerinin hesaplanmasında kullanılmıştır.

3.3.3.1. Hata Kareler Ortalaması (MSE)

MSE ilgili modelin gerçeğe yakın sonuçlar ürettiğini değerlendirmek amacıyla ortalama hata değerlerinin büyüklüğünü ölçer. Bunun için gerçek değerler ile tahmin edilen değerler arasındaki fark ölçülür ve bu farkların karelerinin ortalaması alınır.

Parametre tahmin işlemi sonucunda artık modeller uygulanabilir duruma geldiğinde oluşturulan modellerin başarımlarını ölçmek için hata kareler ortalaması yöntemi kullanılmıştır. Her bir projede, her bir model için hata kareler ortalaması hesaplanmıştır.

Hata kare ortalaması aşağıdaki gibi hesaplanır:

$$HKO = \sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n}$$

y_i = i. gözlem için gerçek değer

$\hat{y}_i$ = i. gözlem için tahmin edilen değer

n = toplam gözlem sayısı

3.3.3.2. Ortalama Bağıl Hata ve Yüzde Bağıl Hata Sapması (MMRE, PRED)

MMRE kestirim doğruluğu için sık kullanılan bir ölçüdür. MMRE bir veri kümesinde göreceli hataların yüzde olarak ortalamasıdır.

PRED(q) model performansı için bir başka önemli ölçüdür. PRED(q), kestirimlerin yüzde kaçının %q hata oranı içinde olduğunu gösterir. Örneğin, PRED(25) = 0.95 ise yapılan kestirimlerin %95'i gerçek değerden maksimum % 25 sapmıştır. Bu

çalışmada PRED değeri hesaplanırken “q” sayısı 25 olarak alınmıştır, çünkü bu sayı endüstride kestirim modelleri için sık kullanılan bir değerdir [38].

Ortalama bağıl hata sayısı aşağıdaki eşitlikten hesaplanır:

$$MRE_i = \left| \frac{ea_i - ee_i}{ea_i} \right|$$

$$MMRE = \frac{1}{n} \sum_{i=1}^{n} MRE_i$$

ea_i = i. gözlemin gerçek değeri

ee_i = i. gözlem için tahmin edilen değer

n = toplam gözlem sayısı

Tahmin değerlerinin yüzde kaçının q değerinden daha düşük hataya sahip olduğunu hesaplamak için $PRED(q)$ kullanılır.

$$PRED(q) = \frac{\lambda}{N}$$

Bu denklemde λ, $MRE_i < q$ değerlerini sağlayan tahminlerin sayısı [39, 40], N ise toplam gözlem sayısıdır.

3.3.3.3. Dengeli Tahmini Bağıl Hata (BPRE)

Yazılım güvenilirlik modellerinin önemli bir karakteristiği de tahmini asimptot doğruluğudur ve bir ürünün kullanıcıya sunmaya hazır olduğunun göstergesidir. Tahmini Bağıl Hata (Predicted Relative Error, PRE) ölçütü ile hesaplanabilir. PRE’nin denklemi aşağıdaki gibidir:

$$PRE_{(i)} = \frac{(ea_i - ee_i)}{ee_i}$$

Fakat PRE, pozitif ve negatif sapmalarda tutarlı bir sonuç vermemektedir, bu nedenle dengeli tahmini bağıl hata (Balanced Predicted Relative Error, BPRE) yöntemi kullanılmaktadır. $BPRE_{(i)}$ değeri aşağıdaki denklem ile hesaplanmaktadır:

$$BPRE_{(i)} = \frac{(ee_i - ea_i)}{ee_i + 2\eta(ea_i - ee_i)}, \quad \eta = \begin{Bmatrix} 0, ee_i > ea_i \\ 1, ee_i < ea_i \end{Bmatrix}$$

$BPRE_{(i)}$ ölçütü +/-1 arasında değer almaktadır. $BPRE_{(i)}$ değeri -/+1 değerine yakın sonuçlar aldığında yüksek tahmin hatası anlamına gelmekte iken 0 değerine yakın sonuç vermesinde ise tahmin edilen toplam hata sayısının gerçek değere yakın olduğu anlaşılmaktadır [19]. Ortalama Dengeli Tahmini Bağıl Hata ise aşağıdaki gibi hesaplanmaktadır:

$$A.BPRE = \frac{\sum_{i=1}^{n} BPRE_{(i)}}{n}$$

4. SONUÇLAR

Bu çalışmanın temel amacı web yazılımı projelerinde, belirlenen yazılım güvenilirlik modellerinin başarımını ölçüp karşılaştırma yapmaktır. Bu amaçla çalışmanın ilk aşamasında modellerin uygulanacağı projeler için hata kayıt verileri elde edilmiştir. Sonrasında kullanılacak yazılım güvenilirlik modelleri seçilmiş ve her bir proje için bu modellerin ortalama değer fonksiyonlarında yer alan parametreler en çok olabilirlik tahmin yöntemiyle tahmin edilmiştir.

Verilerin %100'ü kullanılarak elde edilen parametre tahmin sonuçları Çizelge 7'de verilmiştir.

Çizelge 7 Tahmin Edilen Model Parametreleri (%100)

Projeler / Modeller	Proje A	Proje B	Proje C	Proje D
Üstel Homojen Olmayan Poisson Süreci Goel Okumoto	a =1,6265e+09 b =4,7497e-09	a =287,9912 b =0,0620	a =78,6601 b =0,0885	a =63,4720 b =0,0336
Musa Üstel	a =456,3914 c =784,0297	a =240,3114 c =586,6168	a =78,5387 c =388,6127	a =58,1371 c =160,5720
Büklümlü S Şekilleri	a =417,3858 b =0,1025 β = 9,9600	a =229,9413 b =0,2446 β = 5,7300	a =78,4079 b =0,1417 β = 1,3600	a =56,0866 b =0,1054 β = 9,9400
Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci Modelleri	a =531,4072 b =0,0519	a =235,2945 b =0,1954	a =78,0416 b =0,1839	a =56,5782 b =0,0907
Yamada	a =766,2404 b =0,01 α = 0,0088	a =276,4678 b =0,0682 α =0,0001	a =78,21 b =0,0893 α =0,000124	a =58,7899 b =0,035 α =0,0016
PNZ IFD	a =568,5847 b =0,001 d =0,0136	a =230,3869 b =0,2062 d =-0,01	a =78,0073 b =0,2490 d =0,0646	a =55,8397 b =0,1239 d =0,0336

Proje A için yapılan parametre tahminleri incelendiğinde Üstel Homojen Olmayan Poisson Süreci Goel Okumoto Modelinde b parametresinin oldukça düşük tahmin edildiği görülmektedir. Bu değer projede hata tespit oranının çok düşük olduğunu gösterir. Çok düşük hata tespit oranı projede başlangıçta bulunan hata sayısının yüksek olarak ölçülmesine neden olmuştur. Bu ölçüm a parametresinin çok yüksek olması ile de kanıtlanmaktadır. Üstel Homojen Olmayan Poisson Süreci Goel-

Okumoto Modeli A projesinin henüz olgunlaşmadığını ve devam edecek süreçte bulunan hata sayılarının kümülatif toplamının doğrusala yakın bir şekilde artış göstereceğini tahmin etmektedir. Genel olarak diğer modeller de A projesinde henüz tespit edilmeyen birçok hatanın var olduğunu göstermektedir. Proje için tüm modellerde a parametreleri incelendiğinde 08.11.2015 tarihine kadar bulunan toplam hata sayısına göre başlangıçta yazılımda bulunan hata sayısının çok daha yüksek olarak tahmin edildiği görülmektedir.

Proje B'de tüm modeller için tahmin edilen parametreler incelendiğinde tüm modellerde hataların yaklaşık olarak %80-%90 oranında tespit edildiği şeklinde modelleme yapıldığı görülmektedir. Verilerde de Şekil 4 Proje A Hata Kayıtları incelendiğinde kümülatif hata kayıtlarının zamana göre doğrusal bir şekilde ilerlediği görülmektedir. Kümülatif hatanın doğrusal ilerlemesi projenin henüz olgunlaşmadığını ve başlangıç aşamasında olduğunu gösterir. Bu projenin hata kayıt grafiği ilerleyen dönemde iç bükey ya da S şekline dönüşebilir.

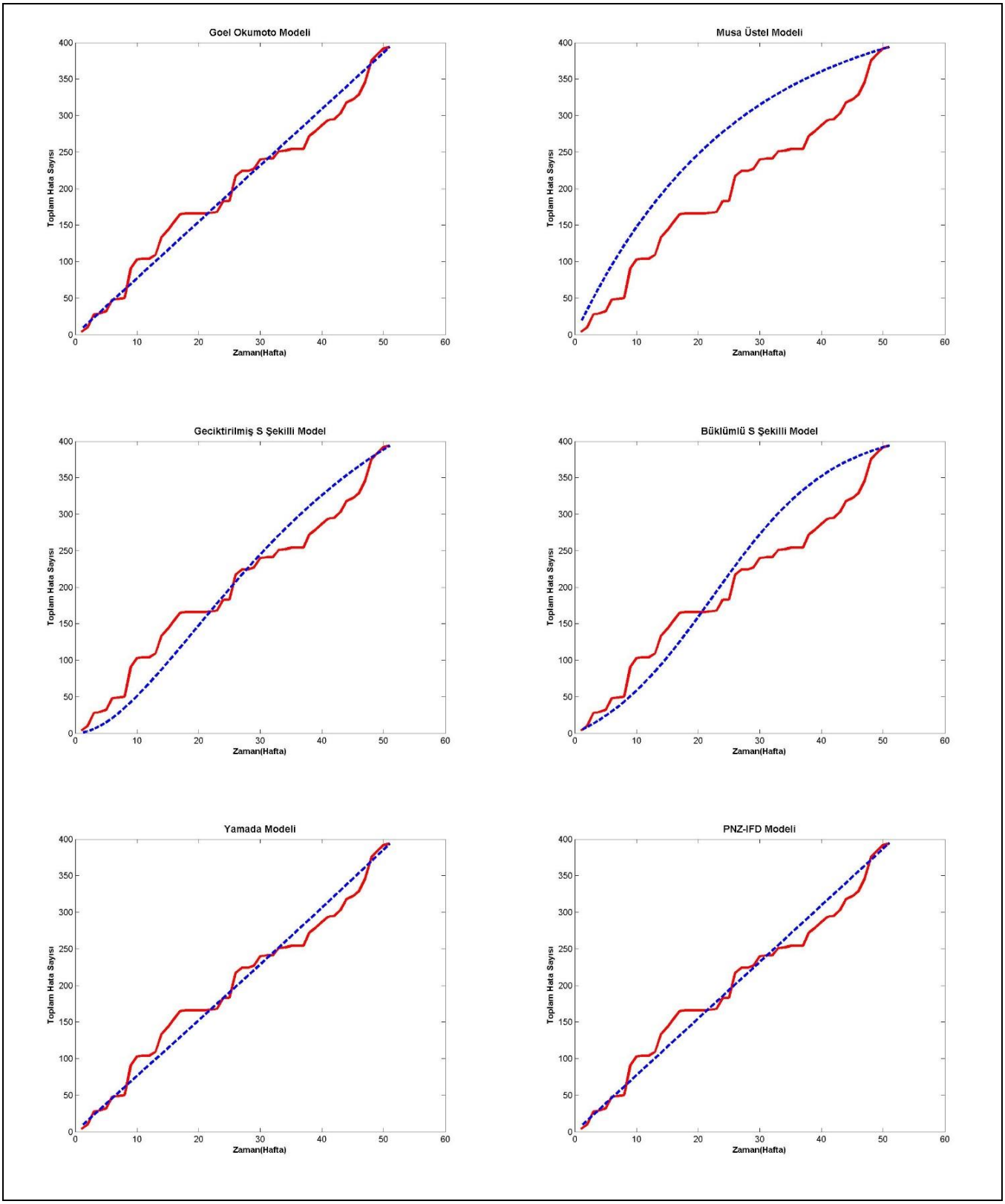
Çizelge 3'te görüldüğü gibi 08.11.2015 tarihine kadar yazılımda 223 adet hata bulunmuştur. Üstel Homojen Olmayan Poisson Süreci Goel Okumoto ve Yamada modelleri yaklaşık olarak başlangıçta sistemde sırası ile 288 ve 276 hatanın bulunduğunu belirtmişlerdir. Yani bu modeller sistemde hala %15-20 oranında hatanın henüz tespit edilemediğini işaret etmektedirler. Diğer modeller ise 229-240 arası başlangıçta var olan hata sayıları ile yazılımda bulunan hataların yaklaşık %90-95 oranında tespit edildiğini belirtmektedirler.

Proje C'de tüm modeller büyük oranda yazılımda bulunan tüm hataların tespit edildiğini işaret etmiştir. Projede 78 hata tespiti yapılmıştır ve tüm modeller projede başlangıçta yaklaşık olarak 78 hatanın bulunduğunu tahmin etmişlerdir. Çizelge 1'de proje C'nin bitmiş olduğu belirtilmiştir. Hata kayıt verilerinin %100'ünü kullanarak yapılan modellemelerin tümü de projenin bitmiş olduğunu ve artık kullanıcıya sunulabilecek güvenilirlikte olduğunu göstermişlerdir.

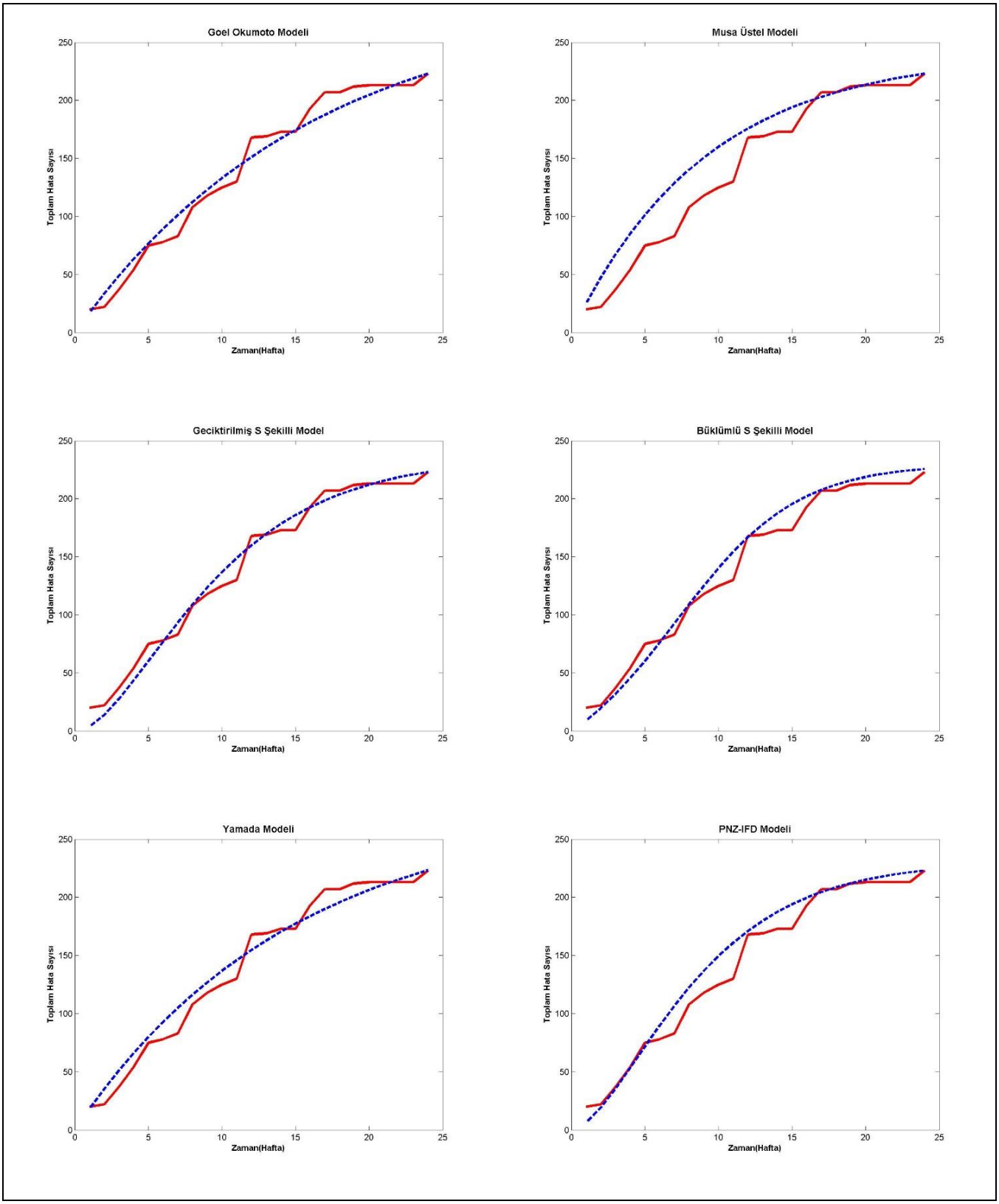
Proje D'de de neredeyse tüm modeller yazılımın belirli bir olgunluğa ulaştığını işaret etmektedir. Çizelge 1'de D projesinin bitmiş olduğu belirtilmişti fakat hata kayıtları incelendiğinde son dönemde yazılımda bozulmaya rastlandığı görülmektedir. Bu da yazılımda hala bozulmanın olabileceğini göstermektedir. Uygulanan modellerin sonuçları incelendiğinde yazılımda halen %10-15 hata bulunduğu anlaşılabilir.

Yazılım bu hali ile kullanıcıya sunulduğunda bozulma olasılığı C projesine göre fazla olduğu tahmin edilmektedir.

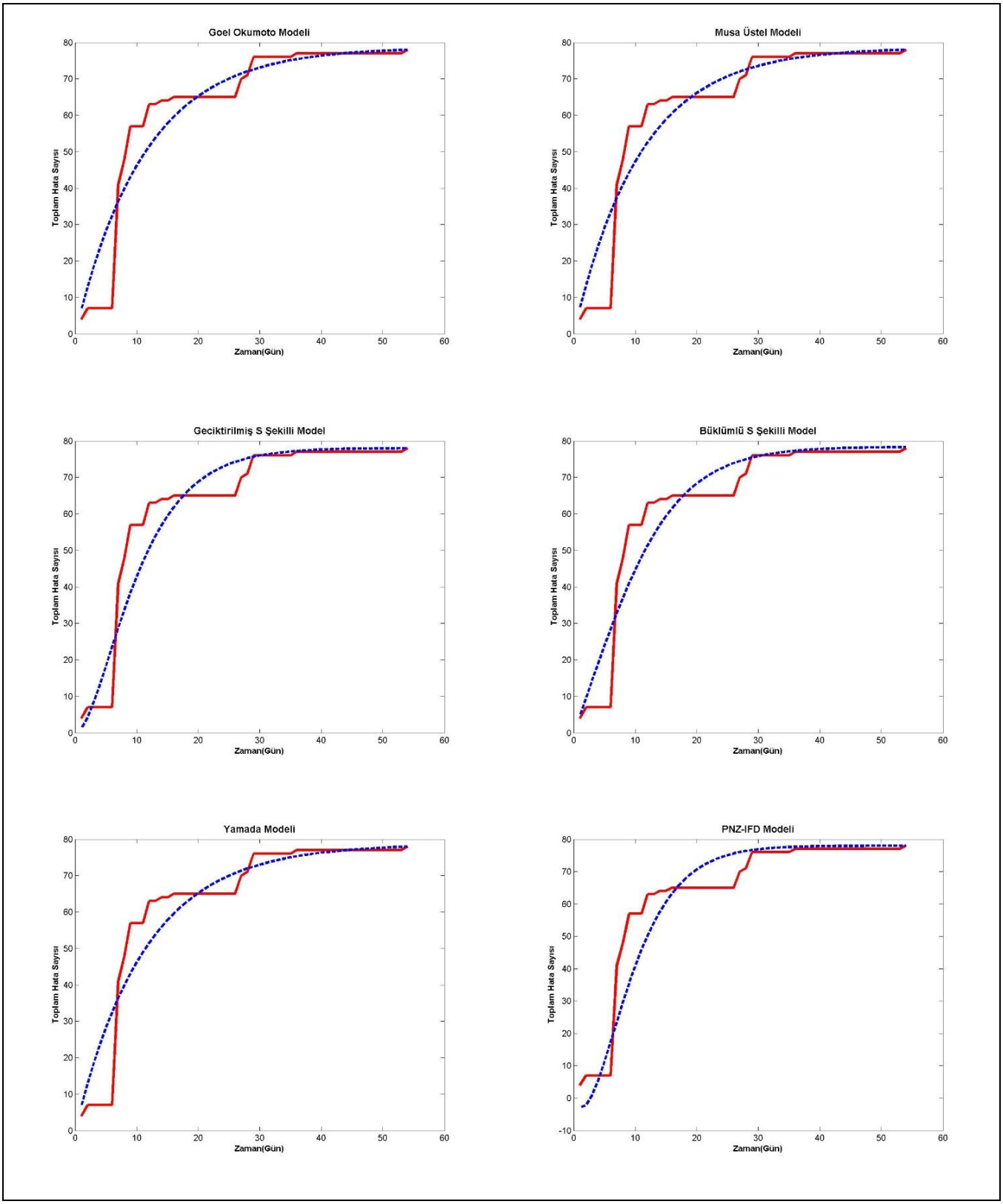
Yazılım güvenilirlik modellerinde bulunan parametrelerin tahmin edilmesiyle bu çalışma kapsamında değerlendirilen yazılım güvenilirlik modellerinin sonuçları her bir proje için alınmıştır. Her bir proje ve model için ortalama değer fonksiyonları hesaplanmış ve grafikleri çıkarılmıştır. Modellerin grafikleri birbirleri ile karıştırılmaması için her model için ayrı grafik oluşturulmuştur. Fakat bir proje için tüm modellerin daha kolay karşılaştırabilmesi için altı ayrı grafik Şekil 8-Şekil 11'de görüldüğü gibi yan yana verilmiştir. Grafiklerde gerçek kümülatif hata verileri düz çizgi ile gösterilirken tahmin model grafikleri kesikli çizgi ile verilmiştir.



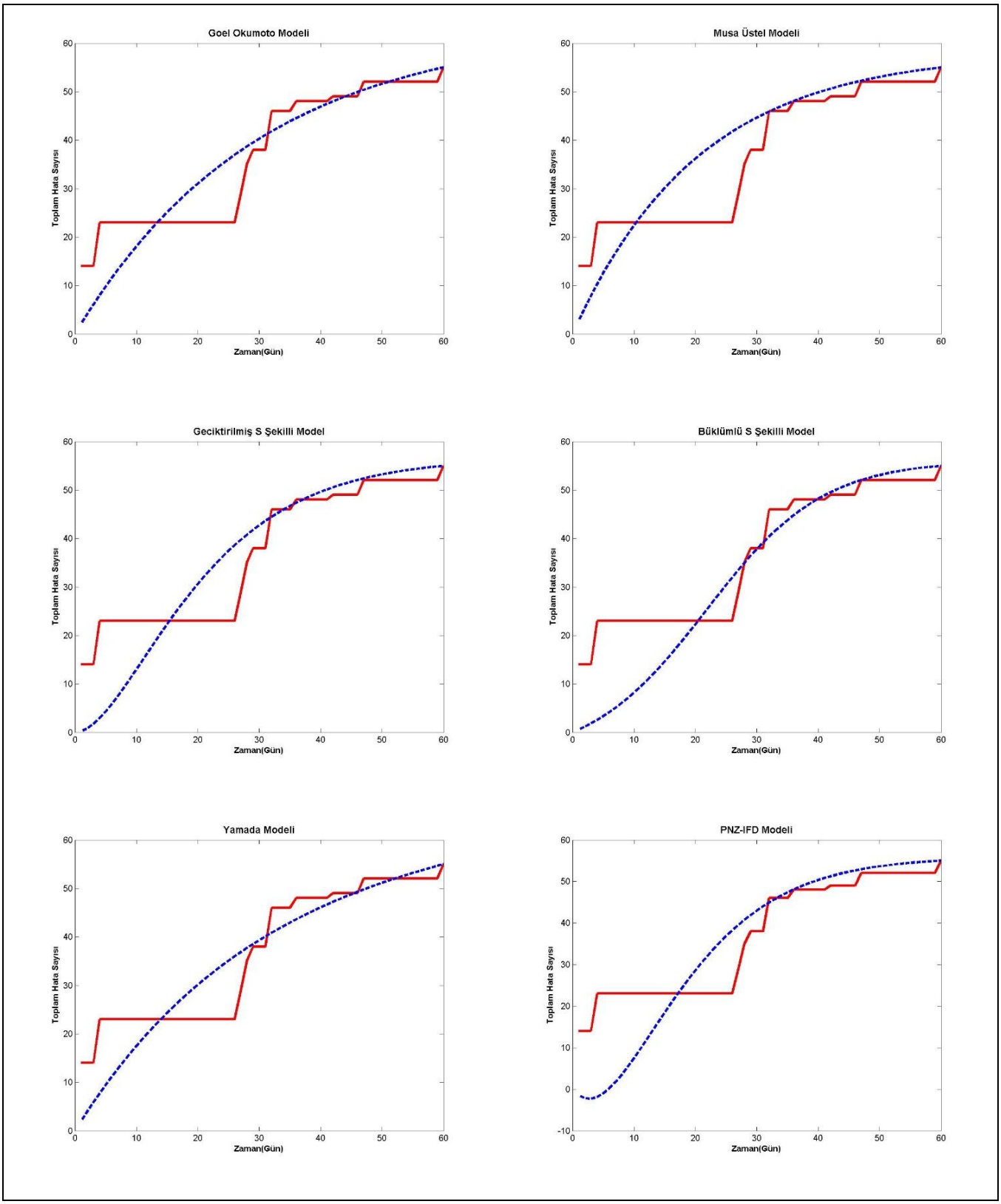
Şekil 8 Proje A için Modellerin Grafikleri (%100)



Şekil 9 Proje B için Modellerin Grafikleri (%100)



Şekil 10 Proje C için Modellerin Grafikleri (%100)



Şekil 11 Proje D için Modellerin Grafikleri (%100)

A projesi için hazırlanan yazılım güvenilirlik modellerinin grafikleri Şekil 8'de verilmiştir. Hata kayıt verilerinden projede tespit edilen hataların kümülatif toplam verisi incelendiğinde zaman içerisinde kümülatif toplam hata sayısının artışında bir azalma görülmemektedir. Kümülatif toplam hata sayısındaki doğrusala yakın artışın devam etmesi bu projenin henüz yeterince olgunlaşmadığını ve içerisinde birçok hatayı barındırdığı şeklinde yorumlanabilir. Model grafikleri incelendiğinde, tüm modellerin doğrusala yakın şekilde artış gösterdiği görülmektedir. Yani tüm modeller devam edecek süreçte birçok hatanın tespit edileceğine işaret etmektedirler.

Grafikler üzerinden modelleri karşılaştırdığımızda Musa Üstel yazılım güvenilirlik modeli hariç diğer modellerin birbirine çok yakın şekilde neredeyse doğrusal bir grafik verdiği görülmektedir. Musa Üstel yazılım güvenilirlik modelinin bu proje için oluşturduğu grafik incelendiğinde, açıkça görülmektedir ki, bu model A projesinin güvenilirliğini modellemede yetersiz kalmıştır. Gerçek değerlerden çok farklı değerleri alan modelin, kullanılan diğer yazılım güvenilirlik modellerine göre daha başarısız kaldığı grafikten de rahatça anlaşılabilmektedir.

B projesi için hazırlanan Şekil 9'da yazılım güvenilirlik modelleri grafikleri incelendiğinde bu projenin A projesine göre daha olgun bir proje olduğu söylenebilir. Projede şimdiye kadar bulunan hataların kümülatif toplamındaki artışı zamanla azalmaktadır. Başka bir deyişle birim zamanda bulunan hata sayısı gün geçtikçe azalmaktadır.

Kullanılan yazılım güvenilirlik modellerinin B projesi için oluşturdukları ortalama değer fonksiyonlarını incelendiğinde de grafiklerde kümülatif hata artışının hız kaybettiği görülmektedir. B projesi için tahmin edilen model parametrelerini yorumlarken özellikle Yamada ve Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modellerinin diğer modellere göre biraz farklılık içerdiğinden bahsedilmişti. Yazılımda içerisinde kalan hata sayılarını belirtmede Yamada ve Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modellerinin ortalama değer fonksiyonlarından da aynı farklılık görülmektedir. Bu iki model için oluşturulan grafiklerde toplam hata sayısının artış hızındaki azalma diğer modellere göre daha azdır. Toplam hata sayısındaki artış hızından Yamada ve Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modellerinin diğer modellere göre bu projede ileride daha fazla hata bulunacağını tahmin ettikleri anlaşılmaktadır.

C projesi için hazırlanan ortalama değer fonksiyonlarını Şekil 10 incelendiğinde, bu projede artık hata oluşumunun neredeyse bittiğinin ve artık yeni hata bulunması arasında cok uzun süreler geçtiği görülmektedir. Tahmin edilen yazılım güvenilirlik model parametrelerini yorumlarken tahmin edilen parametreden, C projesi için tüm modellerin yazılımda bulunan hataların neredeyse hepsinin tespit edildiğini tahmin ettiği belirtilmişti. Şekil 10'da verilen ortalama değer fonksiyonu grafikleri de incelendiğinde de yine aynı sonuçlar görülmektedir. Tüm modeller için oluşturulan ortalama değer fonksiyonları bir süre sonra neredeyse sabit değerler almışlardır.

D projesi için hazırlanan Şekil 11'deki grafikler incelendiğinde, B projesi gibi bu projede de hata oluşumunun gittikçe azaldığı söylenebilir. Hataların kümülatif toplam grafiklerinden bu durum kolayca çıkarılabilmektedir. Her ne kadar D projesinde yeni hata gözlemlenmesi azalmış olsa da, bu azalma C projesi kadar çok değildir. Hala yeni hata tespitlerinin oluşması muhtemeldir. Oluşturulan ortalamada değer fonksiyonlarının grafiklerinde de aynı sonuç çıkarılabilir. Tüm modellerin ortalama değer fonksiyon grafikleri incelendiğinde tüm modellerin birbirine benzer sonuçlar verdiği görülebilir.

Yazılım güvenilirlik modellerinin karşılaştırılması için her model ve proje için oluşturulan ortalama değer fonksiyonlarının MSE, MMRE, PRED ve A.BPRE değerleri hesaplanmıştır.

Çizelge 8 Modellerin Performans Değerleri içerisinden A projesi için MSE ve A.BPRE değerleri incelendiğinde Yamada modelinin diğer modellere göre daha küçük MSE ve A.BPRE değerleri ile modelleme yaptığı görülebilir. A projesi için MMRE değerleri karşılaştırıldığında ise Üstel Homojen Olmayan Poisson Süreci Goel Okumoto Modeli en küçük ortalama bağıl hatası değeri ile modelleme yaptığı görülebilir. A projesinin modellenmesinde PRED değerleri bakımından Üstel Homojen Olmayan Poisson Süreci Goel Okumoto ve PNZ IFD modelleri aynı ve en yüksek değerlere sahip olarak diğer modellerden daha iyi sonuç vermiştir. Çizelge 8'de her bir değerlendirme değerinin üstel değeri o modelin o proje de kaçıncı sırada olduğunu göstermektedir.

Çizelge 8 Modellerin Performans Değerleri (%100)

Projeler / Modeller	Proje A	Proje B	Proje C	Proje D
Üstel Homojen Olmayan Poisson Süreci Goel Okumoto Modeli	MSE=314,2668 [2] MMRE=0,1152 [1] PRED=0,9608 [1] A.BPRE=0,0081 [3]	MSE=113,1534 [3] MMRE=0,0984 [2] PRED=0,9167 [2] A.BPRE=0,0305 [2]	MSE=48,2321 [4] MMRE=0,2673 [3] PRED=0,8889 [2] A.BPRE=0,0460 [4]	MSE=39,4855 [2] MMRE=0,1877 [2] PRED=0,6833 [2] A.BPRE=-0,0040 [1]
Musa Exponential Modeli	MSE=4753 [6] MMRE=0,5444 [6] PRED=0,2157 [6] A.BPRE=0,2972 [6]	MSE=575,9913 [6] MMRE=0,2422 [6] PRED=0,5833 [6] A.BPRE=0,1573 [6]	MSE=47,7816 [3] MMRE=0,275 [4] PRED=0,8889 [4] A.BPRE=0,0538 [6]	MSE=47,7816 [3] MMRE=0,2246 [3] PRED=0,6667 [4] A.BPRE=0,0678 [5]
Büklümlü S Şekilleri Modeli	MSE=1726,6 [5] MMRE=0,1964 [4] PRED=0,7255 [4] A.BPRE=0,0022 [2]	MSE=127,4596 [4] MMRE=0,0919 [1] PRED=0,9583 [1] A.BPRE=0,0019 [1]	MSE=45,0623 [2] MMRE=0,2148 [2] PRED=0,8889 [1] A.BPRE=0,0494 [5]	MSE=67,1216 [5] MMRE=0,2339 [4] PRED=0,6833 [3] A.BPRE=-0,0916 [6]
Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci Modelleri	MSE=936,5059 [4] MMRE=0,2191 [5] PRED=0,6667 [5] A.BPRE=-0,0751 [5]	MSE=83,4796 [1] MMRE=0,1067 [4] PRED=0,8750 [4] A.BPRE=-0,0344 [4]	MSE=42,2765 [1] MMRE=0,1713 [1] PRED=0,8519 [5] A.BPRE=0,0131 [1]	MSE=64,593 [4] MMRE=0,2398 [5] PRED=0,65 [5] A.BPRE=-0,0105 [2]
Yamada Modeli	MSE=311,9062 [1] MMRE=0,1145 [3] PRED= 0,9412 [3] A.BPRE=-0,0005 [1]	MSE=131,4286 [5] MMRE=0,1110 [5] PRED=0,8750 [5] A.BPRE=0,0502 [5]	MSE=49,2921 [5] MMRE=0,2680 [5] PRED=0,8889 [3] A.BPRE=0,0455 [3]	MSE=38,7966 [1] MMRE=0,185 [1] PRED=0,7 [1] A.BPRE=-0,0188 [3]
PNZ IFD	MSE=325,314 [3] MMRE=0,1154 [2] PRED=0,9608 [2] A.BPRE=0,0093 [4]	MSE=91,0526 [2] MMRE=0,1042 [3] PRED=0,9167 [3] A.BPRE=0,0316 [3]	MSE=53,3199 [6] MMRE=0,1831 [6] PRED=0,8333 [6] A.BPRE=-0,0214 [2]	MSE=97,4237 [6] MMRE=0,2884 [5] PRED=0,65 [6] A.BPRE=-0,0379 [4]

B projesi için en küçük MSE değerlerine sahip model Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci modelidir. MMRE, PRED ve A.BPRE değerlerinde ise Büklümlü S Şekilleri Modeli her iki değer için de en iyi sonucu vermiştir.

C projesinde yine B projesi gibi Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci modeli en düşük MSE değeri ile en iyi modellemeyi yapan model olmuştur. Bu projede MMRE ve A.BPRE değerleri bakımından Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci modeli en düşük değere sahip model olmuştur. C projesi için PRED değerleri incelendiğinde Üstel Homojen Olmayan Poisson Süreci Goel Okumoto Modeli, Musa Üstel Modeli, Büklümlü S Şekilleri Modeli ve Yamada Modeli en yüksek ve aynı değerle en iyi sonuç vermiştir, fakat Üstel Homojen Olmayan Poisson Süreci Goel Okumoto Modeli, Musa Üstel Modeli ve Yamada Modeli için hesaplanan MMRE değerleri kabul edilen en küçük MMRE değeri olan 0,25'ten büyük olduğu için PRED değeri bakımından en iyi sonucu Büklümlü S Şekilleri Modelinin verdiği kabul edilir.

D projesi için en düşük MSE ve MMRE değerine ve en büyük PRED değerine sahip model, Yamada modeli iken A.BPRE değeri için Üstel Homojen Olmayan Poisson Süreci Goel Okumoto Modeli en düşük değeri vermiştir.

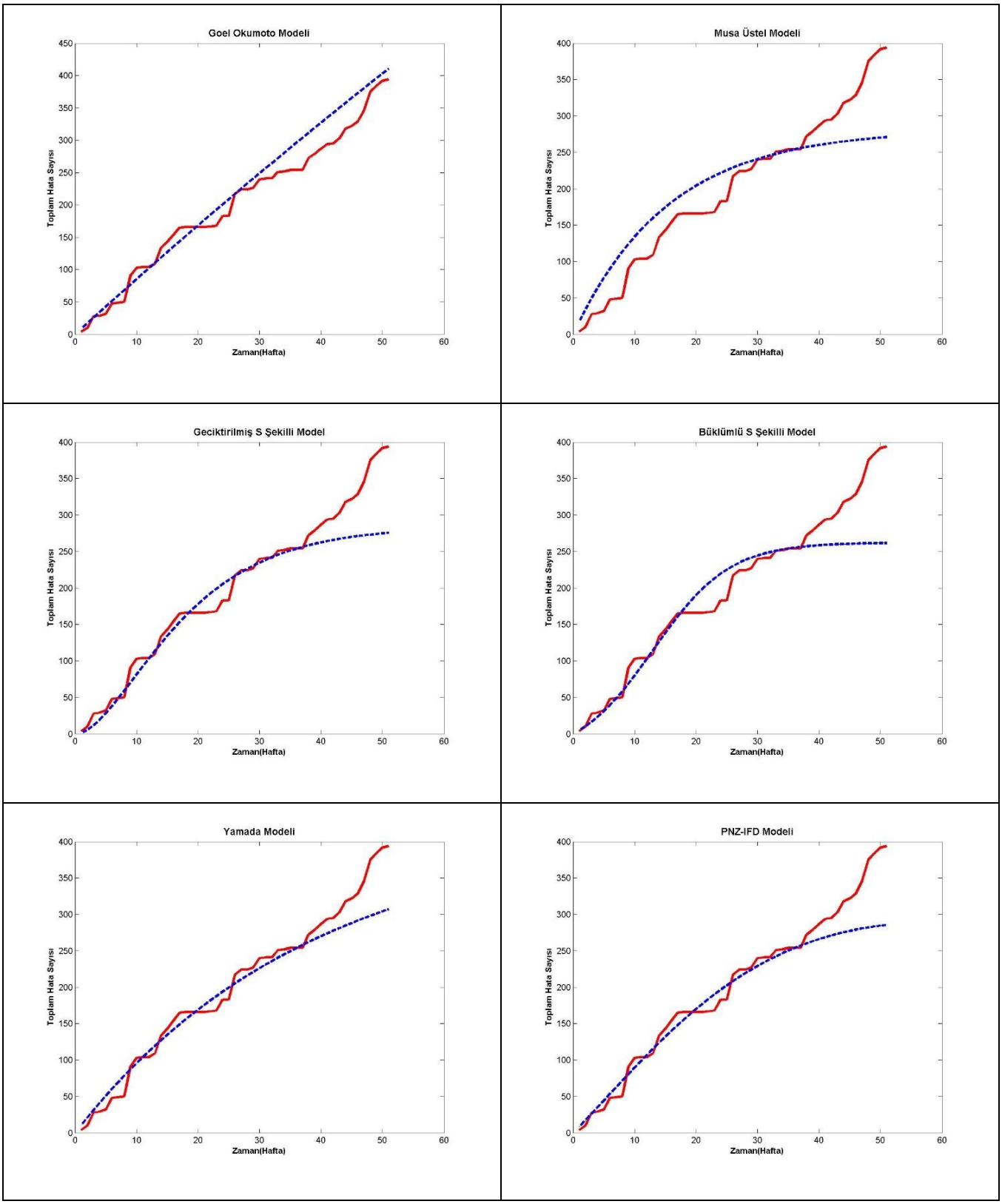
Verilerin %70'i kullanarak elde edilen parametre tahmin sonuçları Çizelge 9 Tahmin Edilen Model Parametreleri (%70)'da verilmiştir.

Çizelge 9 Tahmin Edilen Model Parametreleri (%70)

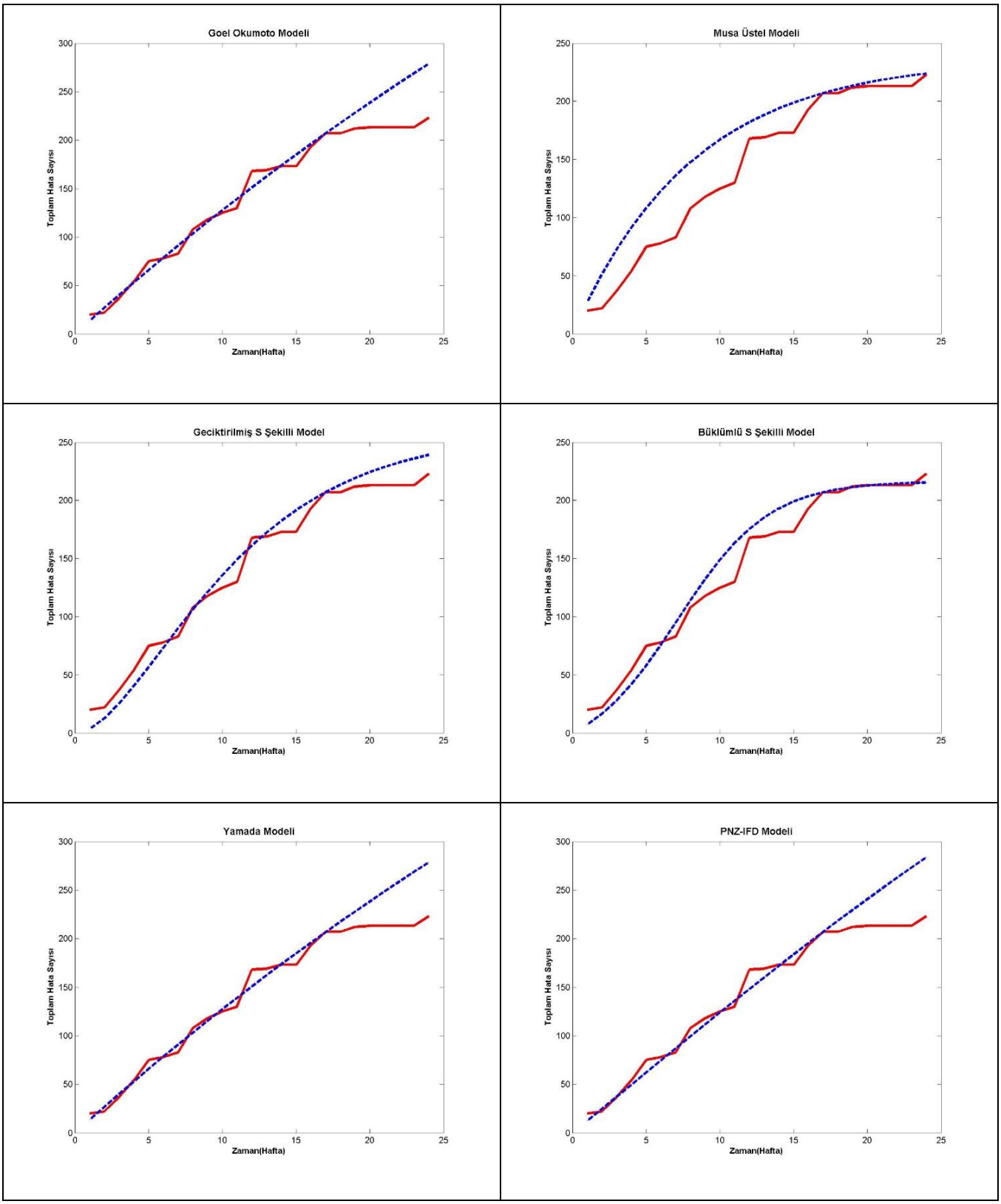
Projeler / Modeller	Proje A	Proje B	Proje C	Proje D
Üstel Homojen Olmayan Poisson Süreci Goel Okumoto	a=439,8380 b=0,0239	a=970,9838 b=0,0141	a=80,3144 b=0,0839	a=67,4027 b=0,0309
Musa Üstel	a=281,34574 c=592,0804	a=236,3742 c=431,6572	a=78,9622 c=284,5075	a=53,225 c=124,1377
Büklümlü S Şekilleri	a=262,3664 b=0,1615 β= 8,19	a=216,4770 b=0,3227 β= 9,9400	a=78,2116 b=0,1724 β= 2,24	a=51,5912 b=0,1272 β= 9,5800
Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci Modelleri	a=283,7191 b=0,1064	a=259,0608 b=0,1758	a=77,4571 b=0,1906	a=52,4353 b=0,1051
Yamada Modeli	a=418,559 b=0,0259 α=0,000026	a=981,4864 b=0,0139 α=0,00002	a=79,967 b=0,0864 α=0,000014	a=74,2837 b=0,0256 α=0,000022
PNZ IFD	a=172.6774 b=0,0372 d=-0,0491	a=131,9857 b=0,0196 d=-0,0938	a=62,8146 b=0,0734 d=-0,0519	a=41,2670 b=0,0497 d=-0,0355

Çizelge 9 Tahmin Edilen Model Parametreleri (%70) verilen parametrelerden A projesi için tahmin edilen parametreleri incelendiğinde, %70 veri ile yapılan tahminlerde a parametresinin genel olarak tüm modellerde düşüş yaşadığı görülebilir. Bu durum tüm modellerin %70 veri ile daha önceki bir zaman için yapılan tahminlemede daha az başlangıç hatası tahmin ettiğini gösterir. B projesinde Üstel Homojen Olmayan Poisson Süreci Goel Okumoto ve Yamada modellerinde a parametresinin artış gösterdiği PNZ IFD modelinde ise azaldığı diğer modellerde ise yaklaşık aynı değerde kaldığı gözlemlenmektedir. C ve D projesinde tahmin edilen parametrelerin %100 verinin kullanıldığı durum ile benzer olduğu anlaşılabilir. Bu durum C ve D projelerinin eldeki ilk %70 verilerinin kullanıldığında, projelerin olgunlaştığını göstermektedir.

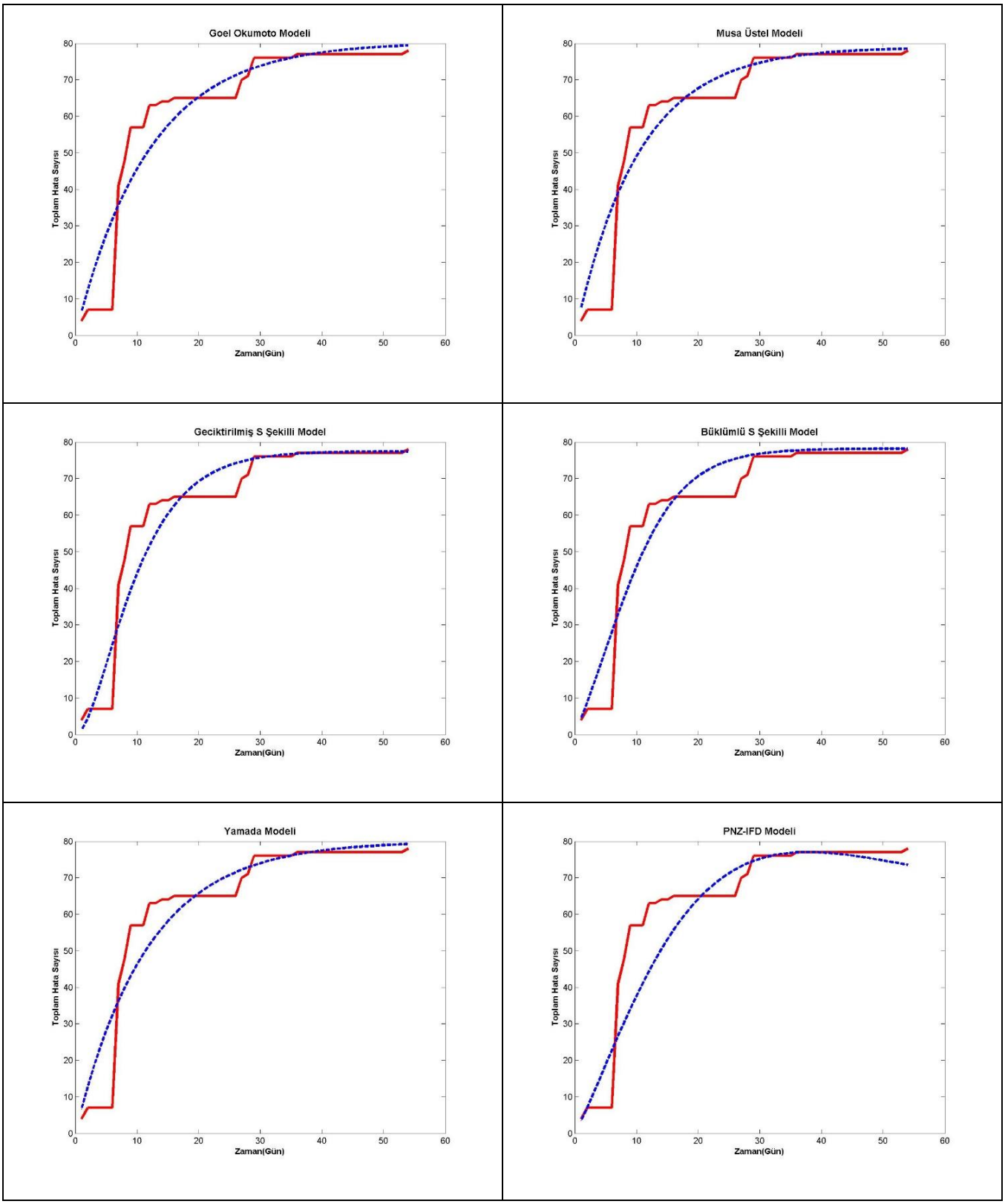
A projesi için verilerin ilk %70'i kullanılarak elde edilen modellerin grafikleri Şekil 12 Proje A için Modellerin Grafikleri (%70)'de verilmiştir.



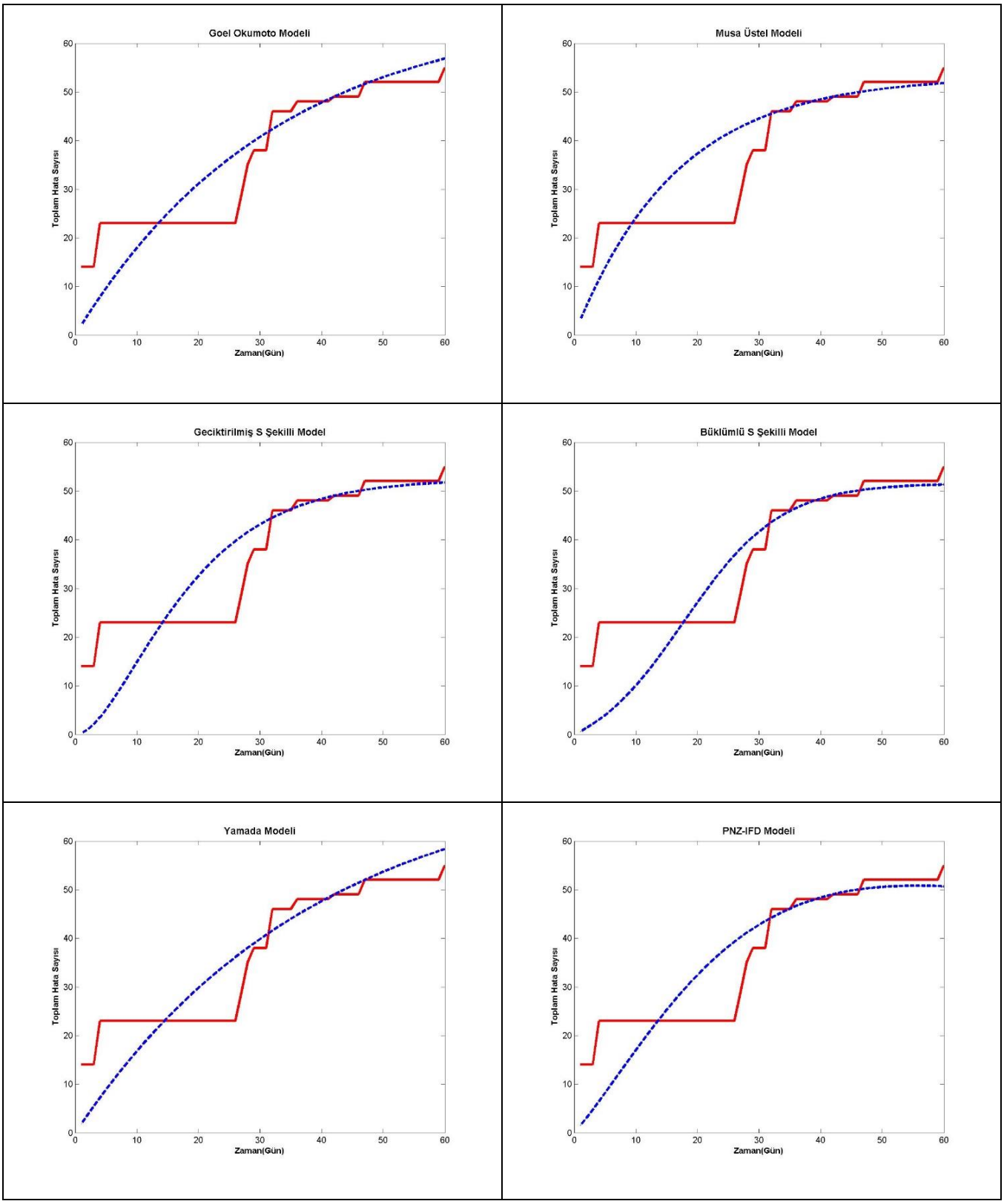
Şekil 12 Proje A için Modellerin Grafikleri (%70)



Şekil 13 Proje B için Modellerin Grafikleri (%70)



Şekil 14 Proje C için Modellerin Grafikleri (%70)



Şekil 15 Proje D için Modellerin Grafikleri (%70)

A projesi için hazırlanan modellerinin grafikleri Şekil 12'de verilmiştir. Hata kayıt verilerinin ilk %70'i incelendiğinde kümülatif hata artışının iç bükey bir şekil aldığı görülmektedir. Modeller bu veriyi kullandıkları için iç bükey bir şekilde model tahmini yapmışlardır. Sadece Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modeli doğrusal bir grafik izlemiştir.

B projesinin ilk %70 verisi ile oluşturulan kümülatif hata grafiğinin doğrusala yakın olduğu anlaşılmaktadır. B projesinin ilk %70 verisi kullanılarak yapılan modellemelerde de Yamada, PNZ IFD ve Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modellerinin doğrusala yakın bir grafiği izledikleri görülebilir. Diğer modeller iç bükey bir grafik izlemişlerdir. B projesinde ilk %70 verisinden sonraki bölümde bükülme yaşadığı için iç bükey modelleme yapan modellerin daha iyi sonuç vereceği düşünülmektedir.

C projesinde verilerin %70'i kullanıldığı durumda, verilerin %100'ü kullanıldığı durumdaki gibi grafiğin iç bükey olduğu görülebilir. Bu proje için tüm modellerin grafikleri de iç bükeydir.

D projesi için verilerin ilk %70'inin kullanılarak hazırlanan modellerin grafikleri incelendiğinde verilerin ilk %70'inin iç bükey olduğu ve bu nedenle güvenilirlik modellerinin de iç bükey olacak şekilde modellendiği görülmektedir.

Çizelge 10 Modellerin Performans Değerleri (%70) incelendiğinde, A projesi için MSE, MMRE ve A.BPRE değerlerinde en düşük değere, PRED değeri için ise en büyük değere Üstel Homojen Olmayan Poisson Süreci Goel Okumoto Modeli sahip olduğu için Üstel Homojen Olmayan Poisson Süreci Goel Okumoto Modelinin diğer modellerden daha iyi modelleme yaptığı düşünülmektedir.

B projesi için ise PRED değeri en büyük değere Üstel Homojen Olmayan Poisson Süreci Goel Okumoto Modelinde, Musa Üstel Modelinde, Büklümlü S Şekilleri Modellerinde ve Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci Modellerinde görülmektedir. Ancak B projesi için MSE, MMRE ve A.BPRE değerlerinde en düşük değere Büklümlü S Şekilleri Modelleri sahip olup için Büklümlü S Şekilleri Modellerinin diğer modellerden daha iyi modelleme yaptığı düşünülmektedir.

Çizelge 10 Modellerin Performans Değerleri (%70)

Projeler / Modeller	Proje A	Proje B	Proje C	Proje D
Üstel Homojen Olmayan Poisson Süreci Goel Okumoto Modeli	MSE= 2236,8 [1] MMRE= 0,1085 [1] PRED=1 [1] A.BPRE=-0,0929 [1]	MSE= 1527,8 [5] MMRE= 0,1638 [5] PRED=1 [5] A.BPRE=0,1367 [5]	MSE= 2,7 [5] MMRE= 0,0195 [5] PRED=1 [5] A.BPRE=0,0191 [6]	MSE= 5,9 [5] MMRE= 0,0396 [5] PRED=1 [5] A.BPRE=0,0366 [5]
Musa Üstel Modeli	MSE=5011,4 [5] MMRE=0,1691 [5] PRED=0,7333 [5] A.BPRE=-0,1380 [5]	MSE=27,8 [2] MMRE=0,0208 [2] PRED=1 [2] A.BPRE=0,0201 [2]	MSE=1,07 [2] MMRE=0,0124 [2] PRED=1 [2] A.BPRE=0,0121 [2]	MSE=1,6 [2] MMRE=0,0199 [2] PRED=1 [2] A.BPRE=-0,0141 [2]
Büklümlü S Şekilleri Modelleri	MSE=5809 [6] MMRE=0,1823 [6] PRED=0,7333 [6] A.BPRE=-0,1466 [6]	MSE=10,2 [1] MMRE=0,0101 [1] PRED=1 [1] A.BPRE=-0,0006 [1]	MSE=1,1 [3] MMRE=0,0135 [3] PRED=1 [3] A.BPRE=0,0132 [3]	MSE=1,9 [3] MMRE=0,0226 [3] PRED=1 [3] A.BPRE=-0,0153 [3]
Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci Modelleri	MSE=4543 [4] MMRE=0,1594 [4] PRED=0,7333 [4] A.BPRE=-0,1308 [4]	MSE= 236,0 [3] MMRE=0,0665 [3] PRED=1 [3] A.BPRE=0,0618 [3]	MSE=1 [1] MMRE=0,0042 [1] PRED=1 [1] A.BPRE=0,0032 [1]	MSE=1,5 [1] MMRE=0,0194 [1] PRED=1 [1] A.BPRE=-0,0130 [1]
Yamada Modeli	MSE=2403,5 [2] MMRE=0,1129 [2] PRED= 1 [2] A.BPRE=-0,0958 [2]	MSE=1502 [4] MMRE=0,1620 [4] PRED=0,8571 [4] A.BPRE=0,1355 [4]	MSE=2,2616 [4] MMRE=0,018 [4] PRED=1 [4] A.BPRE=0,0176 [4]	MSE=10,674 [6] MMRE=0,0535 [6] PRED=1 [6] A.BPRE=0,0498 [6]
PNZ IFD	MSE=3701,8 [3] MMRE=0,1415 [3] PRED=0,8 [3] A.BPRE=-0,1174 [3]	MSE=1780,7 [6] MMRE=0,1767 [6] PRED=0,7143 [6] A.BPRE=0,1458 [6]	MSE=3,8920 [6] MMRE=0,0197 [6] PRED=1 [6] A.BPRE=-0,0190 [5]	MSE=2,5197 [4] MMRE=0,0263 [4] PRED=1 [4] A.BPRE=-0,0195 [4]

C projesi için en düşük MSE, MMRE ve A.BPRE değeri ve en büyük PRED değeri Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci Modellerinde görülmektedir ve bu modelin C projesini en iyi şekilde modellediği söylenebilir.

D projesi için de C projesindeki gibi en düşük MSE, MMRE ve A.BPRE değerleri ve en büyük PRED değeri Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci Modellerinde görülmektedir ve bu modelin de D projesini en iyi şekilde modellediği düşünülmektedir.

Verilerin %50'si kullanarak elde edilen parametre tahmin sonuçları Çizelge 11 Tahmin Edilen Model Parametreleri (%50)'de verilmiştir.

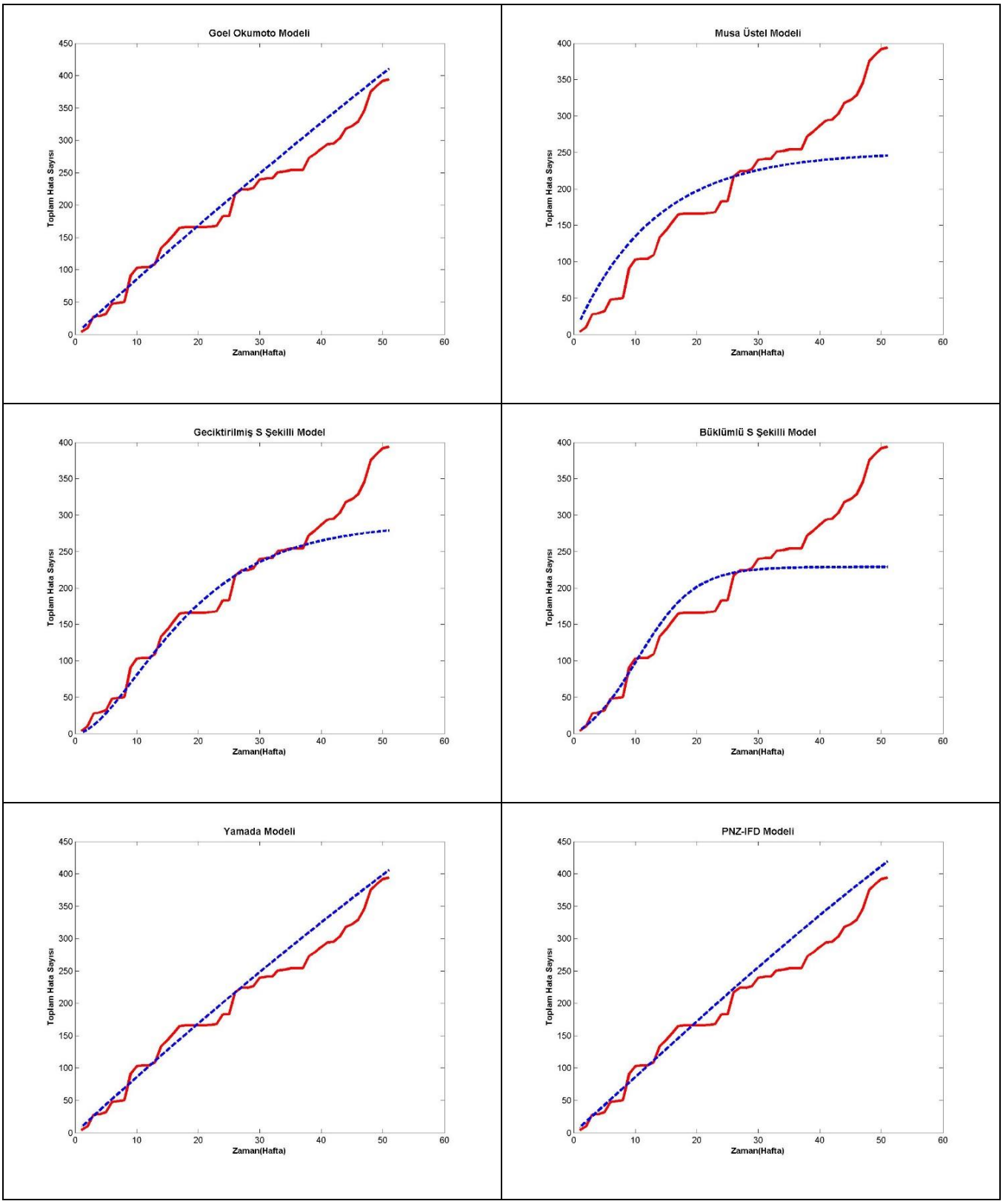
Çizelge 11 Tahmin Edilen Model Parametreleri (%50)

Projeler / Modeller	Proje A	Proje B	Proje C	Proje D
Üstel Homojen Olmayan Poisson Süreci Goel-Okumoto	$a=2958{,}697$ $b=0{,}00293$	$a=4863{,}1588$ $b=0{,}00293$	$a=76{,}595$ $b=0{,}0908$	$a=51{,}882$ $b=0{,}0439$
Musa Üstel	$a=250{,}66$ $c=435{,}65$	$a=198{,}73$ $c=313{,}59$	$a=72{,}90$ $c=225{,}63$	$a=41{,}233$ $c=96{,}73$
Büklümlü S Şekilleri	$a=228{,}911$ $b=0{,}2184$ $\beta=9{,}66$	$a=179{,}195$ $b=0{,}4571$ $\beta=0{,}46$	$a=70{,}122$ $b=0{,}3377$ $\beta=9{,}99$	$a=41{,}2073$ $b=0{,}175$ $\beta=9{,}9$
Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci	$a=288{,}04$ $b=0{,}10427$	$a=271{,}5965$ $b=0{,}17441$	$a=71{,}057$ $b=0{,}2288$	$a=40{,}622$ $b=0{,}1477$
Yamada Modeli	$a=2194$ $b=0{,}004$ $\alpha=0{,}00009$	$a=571{,}6172$ $b=0{,}029$ $\alpha=0{,}000001$	$a=75{,}5562$ $b=0{,}0965$ $\alpha=0{,}000015$	$a=67{,}4602$ $b=0{,}0276$ $\alpha=0{,}00002$
PNZ IFD	$a=277{,}0184$ $b=0{,}01$ $d=-0{,}0307$	$a=296{,}1905$ $b=0{,}11$ $d=-0{,}0287$	$a=71{,}1162$ $b=0{,}2129$ $d=0{,}01$	$a=25{,}3371$ $b=0{,}001$ $d=-0{,}05$

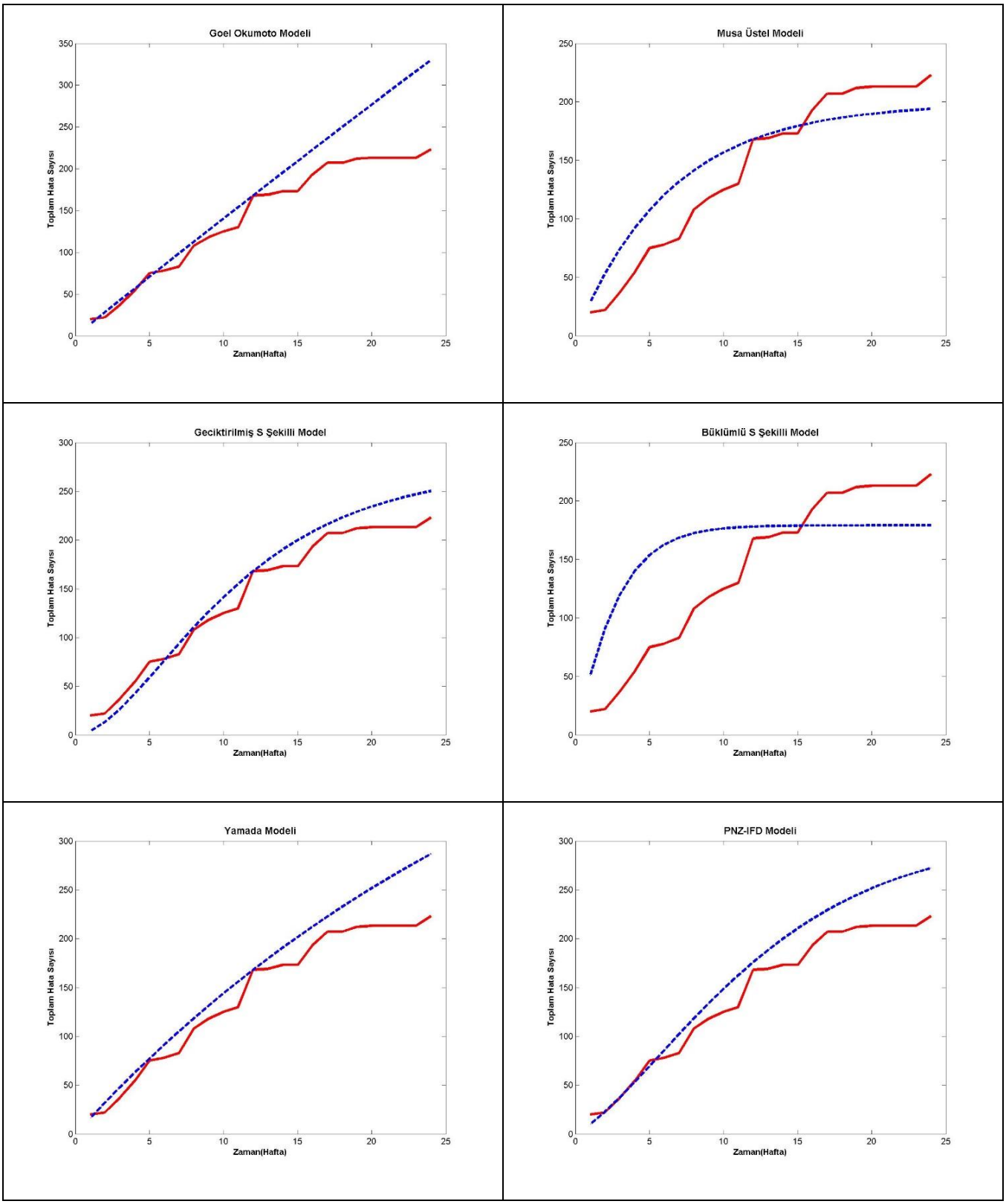
Çizelge 11'de verilen parametrelerden A projesi için tahmin edilen parametreleri incelendiğinde, ilk %50 veri ile yapılan tahminlerde a parametresinin genel olarak düşüş yaşadığı yalnızca Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modeli ve Yamada modelinde a parametresinin arttığı görülebilir. Bu durum bu modellerin %50 veri ile yapılan tahminlemede daha fazla başlangıç hatası tahmin ettiğini gösterir. B projesinde Üstel Homojen Olmayan Poisson Süreci Goel

Okumoto ve Yamada modellerinde a parametresinin artış gösterdiği diğer modellerde ise yaklaşık aynı değerde kaldığı gözlemlenmektedir. C projesinde tahmin edilen a parametrelerinin %100 verinin kullanıldığı duruma göre az da olsa düşüş yaşadıkları görülebilir. D projesinde tahmin edilen a parametrelerinin %100 verinin kullanıldığı duruma göre az miktarda düşüş yaşadıkları görülebilir. Yalnızca PNZ IFD modelinde ciddi bir düşüş gözlemlenmektedir.

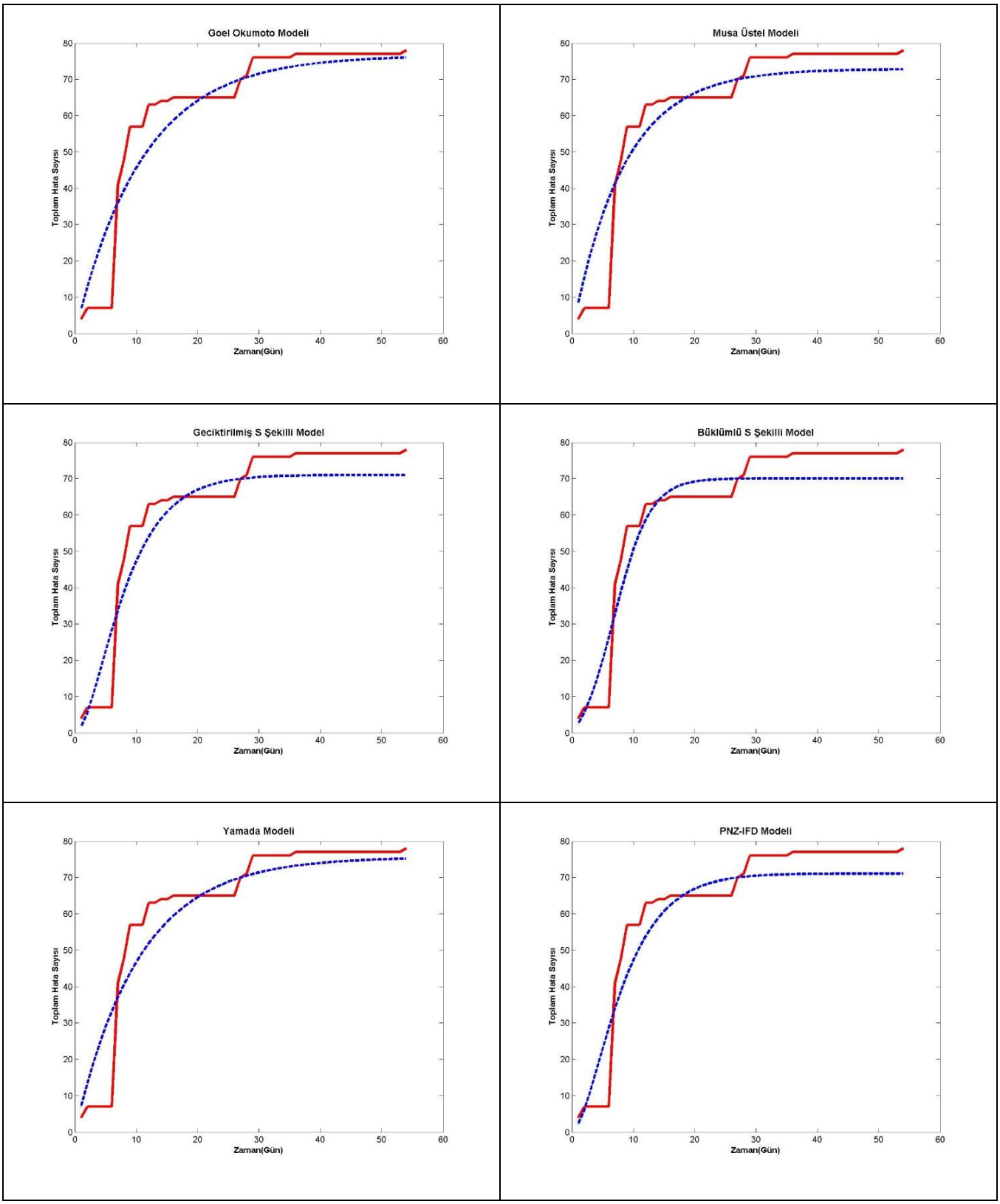
A projesi için verilerin ilk %50'i kullanılarak elde edilen modellerin grafikleri Şekil 16 Proje A için Modellerin Grafikleri (%50) Şekil 17 Proje B için Modellerin Grafikleri (%50), Şekil 18 Proje C için Modellerin Grafikleri (%50) ve Şekil 19 Proje D için Modellerin Grafikleri (%50)'da verilmiştir.



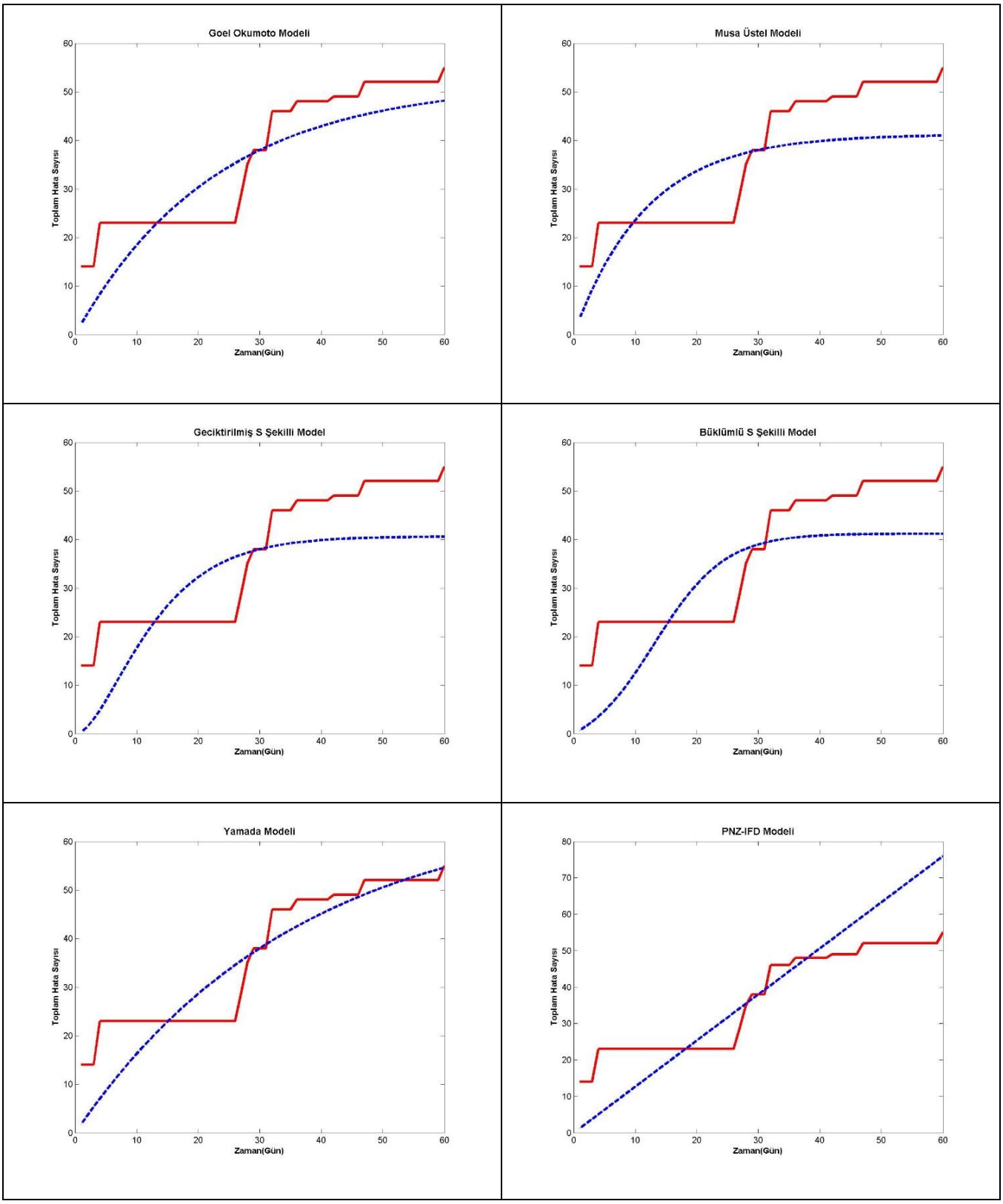
Şekil 16 Proje A için Modellerin Grafikleri (%50)



Şekil 17 Proje B için Modellerin Grafikleri (%50)



Şekil 18 Proje C için Modellerin Grafikleri (%50)



Şekil 19 Proje D için Modellerin Grafikleri (%50)

A projesinde hata kayıt verilerinin ilk %50'si incelendiğinde kümülatif hata artışının doğrusal olduğu görülmektedir. Bu projede Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modeli, Yamada modeli ve PNZ IFD modeli doğrusal bir modelleme yapmışlardır. Diğer modeller ise iç bükey bir modelleme yapmıştır. Bu modellerin daha kötü sonuç vereceği söylenebilir.

B projesinde hata kayıt verilerinin ilk %50'si incelendiğinde kümülatif hata artışının doğrusal olduğu fakat daha sonra iç bükey bir grafik izlediği görülmektedir. Bu projede Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modeli ve Yamada modeli doğrusal bir modelleme yapmışlardır. Diğer modeller ise iç bükey bir modelleme yapmıştır. Proje kümülatif hata toplam verisinin %50 veriden sonra kırılıp iç bükey grafik izlemesi nedeni ile doğrusal modelleme yapan modellerin daha kötü sonuç vereceği düşünülebilir.

C projesinde verilerin %50'si kullanıldığı durumda, verilerin %100'ü kullanıldığı durumdaki gibi grafiğin iç bükey olduğu görülebilir. Bu proje için tüm modellerin grafikleri de iç bükeydir.

D projesi için verilerin ilk %50'sinin kullanılarak hazırlanan modellerin grafikleri incelendiğinde PNZ IFD modeli hariç tüm modellerin iç bükey bir grafik izlediği görülebilir.

Çizelge 12'de modellerin performans değerleri incelendiğinde, A projesi için MSE ve MMRE en düşük değere Yamada modelinde sahiptir. En düşük A.BPRE değerine ise Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci Modellerinde sahiptir. PRED değeri için en büyük değere Üstel Homojen Olmayan Poisson Süreci Goel Okumoto, Yamada Modeli ve PNZ IFD modellerinde sahiptir.

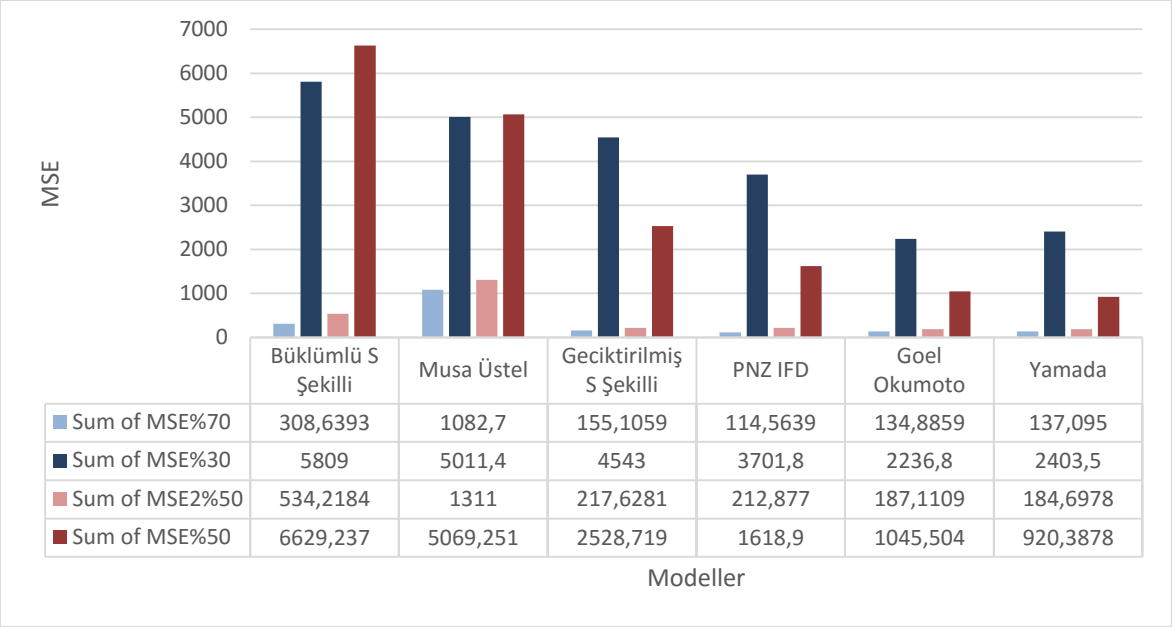
B projesi için ise MSE, MMRE ve A.BPRE değerleri en küçük değerine Musa Üstel modelinde sahip iken PRED değerinin en büyük değeri, Musa Üstel Modelinde, Büklümlü S Şekilleri Modellerinde ve Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci Modellerinde görülmektedir. Dolayısı ile Musa Üstel modelinin B projesinin ilk %50 verisi kullanıldığında en iyi modellemeye yapan model olduğu düşünülmektedir.

Çizelge 12 Modellerin Performans Değerleri (%50)

Projeler / Modeller	Proje A	Proje B	Proje C	Proje D
Üstel Homojen Olmayan Poisson Süreci Goel Okumoto	MSE=1045,504 [2] MMRE= 0,1007 [3] PRED=1 [2] A.BPRE=0,0895 [3]	MSE= 4056,722 [6] MMRE=0,2677 [6] PRED=0,583 [6] A.BPRE=0,2024 [6]	MSE= 6,5195 [1] MMRE= 0,0301 [1] PRED=1 [1] A.BPRE=-0,0291 [1]	MSE= 28,8866 [2] MMRE= 0,1053 [2] PRED=1 [2] A.BPRE=-0,0940 [2]
Musa Üstel	MSE=5069,251 [5] MMRE=0,1638 [5] PRED=0,76 [5] A.BPRE=-0,1322 [5]	MSE=360,863 [1] MMRE=0,0817 [1] PRED=1 [1] A.BPRE=-0,0621 [1]	MSE=20,61 [3] MMRE=0,0582 [3] PRED=1 [3] A.BPRE=-0,0549 [3]	MSE=95,043 [4] MMRE=0,1878 [4] PRED=0,967 [4] A.BPRE=-0,1565 [5]
Büklümlü S Şekilleri	MSE=6639,237 [6] MMRE=0,1926 [6] PRED=0,68 [6] A.BPRE=-0,1501 [6]	MSE=797,818 [3] MMRE=0,1163 [3] PRED=1 [3] A.BPRE=-0,086 [2]	MSE=42,883 [6] MMRE=0,0838 [6] PRED=1 [6] A.BPRE=-0,0772 [6]	MSE=85,576 [3] MMRE=0,1773 [3] PRED=0,967 [3] A.BPRE=-0,1457 [4]
Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci Modelleri	MSE=2528,719 [4] MMRE=0,095 [2] PRED=0,84 [4] A.BPRE=-0,0747 [1]	MSE= 494,7153 [2] MMRE=0,1037 [2] PRED=1 [2] A.BPRE=0,0930 [3]	MSE=33,226 [5] MMRE=0,0739 [5] PRED=1 [5] A.BPRE=-0,0686 [5]	MSE=98,245 [5] MMRE=0,1904 [5] PRED=0,967 [5] A.BPRE=-0,1582 [6]
Yamada Modeli	MSE=920,3878 [1] MMRE=0,0939 [1] PRED= 1 [1] A.BPRE=0,0837 [2]	MSE=1557,2 [5] MMRE=0,1694 [4] PRED=0,75 [5] A.BPRE=0,1408 [4]	MSE=9,0438 [2] MMRE=0,0372 [2] PRED=1 [2] A.BPRE=-0,0357 [2]	MSE=8,7466 [1] MMRE=0,0494 [1] PRED=1 [1] A.BPRE=-0,0348 [1]
PNZ IFD	MSE=1618,9 [3] MMRE=0,13 [4] PRED=1 [3] A.BPRE=0,1129 [4]	MSE=1423,8 [4] MMRE=0,1779 [5] PRED=0,9167 [4] A.BPRE=0,1496 [5]	MSE=32,6652 [4] MMRE=0,0733 [4] PRED=1 [4] A.BPRE=-0,0681 [4]	MSE=133,9964 [6] MMRE=0,1808 [6] PRED=0,7 [6] A.BPRE=0,1193 [3]

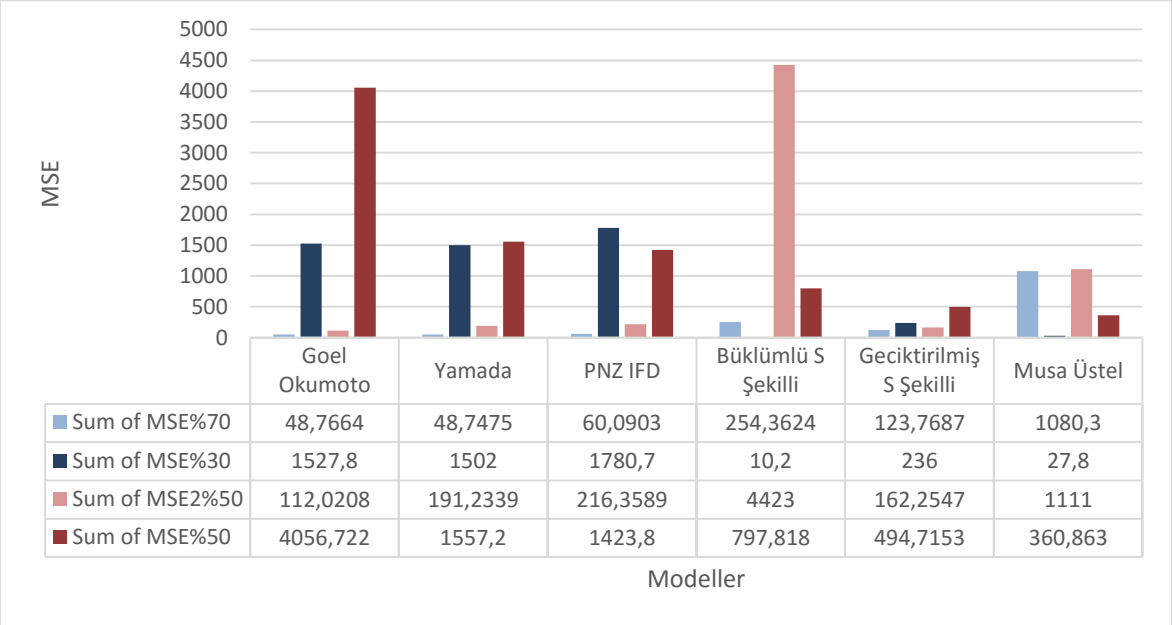
C projesi için PRED değeri her modelde aynı ve bir değerine sahip olmasına rağmen Üstel Homojen Olmayan Poisson Süreci Goel Okumoto da MSE, MMRE ve A.BPRE değerleri en küçük değere sahip olduğu için Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modelinin B projesini en iyi modelleyeceği söylenebilir.

D projesinde ise MSE, MMRE ve A.BPRE değerleri en düşük değere Yamada modelinde ulaşmıştır. PRED değeri ise Üstel Homojen Olmayan Poisson Süreci Goel Okumoto ve Yamada Modelinde en büyük değere sahiptir. Bu durumda D projesi için en iyi modellemenin Yamada modeli ile yapılabileceği düşünülmektedir.



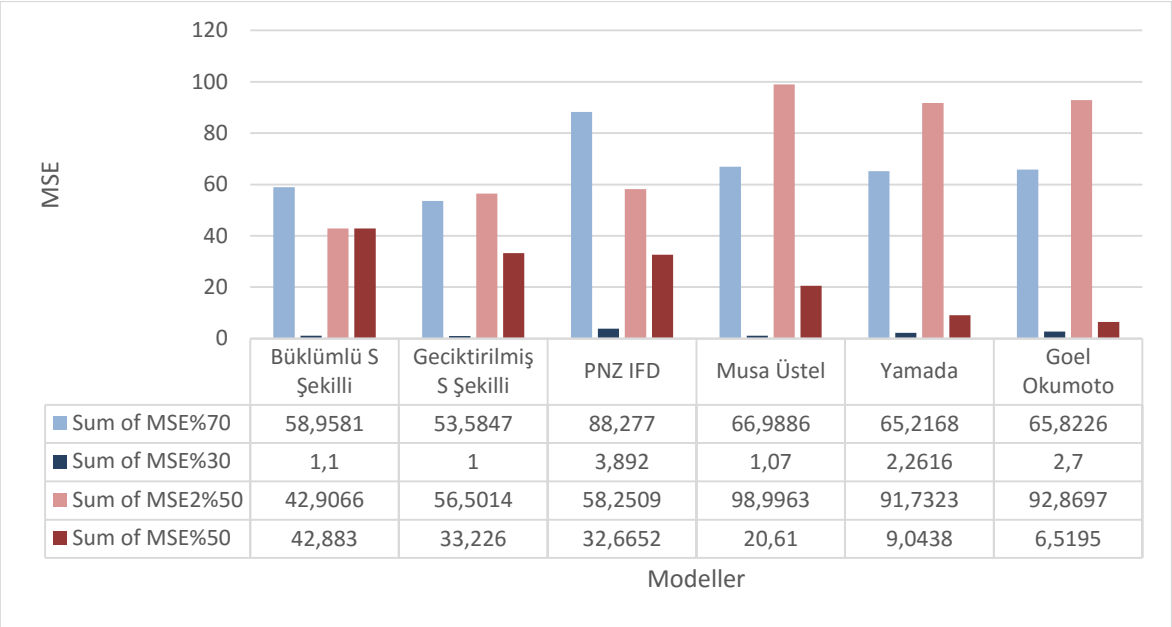
	Büklümlü S Şekilli	Musa Üstel	Geciktirilmiş S Şekilli	PNZ IFD	Goel Okumoto	Yamada
Sum of MSE%70	308,6393	1082,7	155,1059	114,5639	134,8859	137,095
Sum of MSE%30	5809	5011,4	4543	3701,8	2236,8	2403,5
Sum of MSE2%50	534,2184	1311	217,6281	212,877	187,1109	184,6978
Sum of MSE%50	6629,237	5069,251	2528,719	1618,9	1045,504	920,3878

Şekil 20 Proje A Ortalama Hata Değerleri



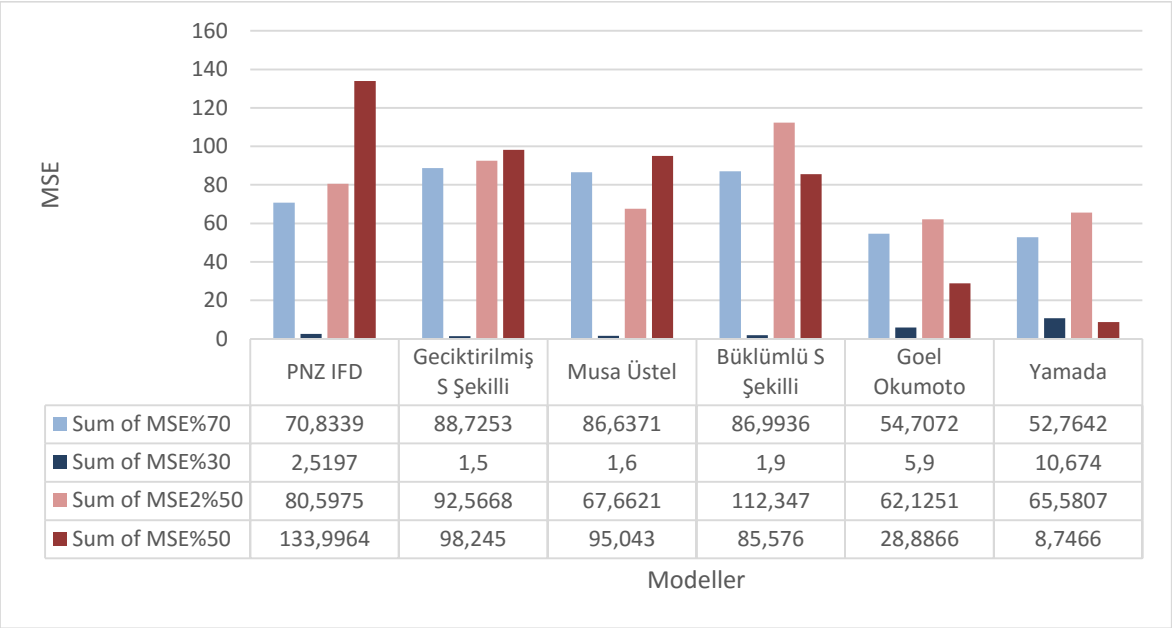
	Goel Okumoto	Yamada	PNZ IFD	Büklümlü S Şekilli	Geciktirilmiş S Şekilli	Musa Üstel
Sum of MSE%70	48,7664	48,7475	60,0903	254,3624	123,7687	1080,3
Sum of MSE%30	1527,8	1502	1780,7	10,2	236	27,8
Sum of MSE2%50	112,0208	191,2339	216,3589	4423	162,2547	1111
Sum of MSE%50	4056,722	1557,2	1423,8	797,818	494,7153	360,863

Şekil 21 Proje B Ortalama Hata Değerleri



	Büklümlü S Şekilli	Geciktirilmiş S Şekilli	PNZ IFD	Musa Üstel	Yamada	Goel Okumoto
Sum of MSE%70	58,9581	53,5847	88,277	66,9886	65,2168	65,8226
Sum of MSE%30	1,1	1	3,892	1,07	2,2616	2,7
Sum of MSE2%50	42,9066	56,5014	58,2509	98,9963	91,7323	92,8697
Sum of MSE%50	42,883	33,226	32,6652	20,61	9,0438	6,5195

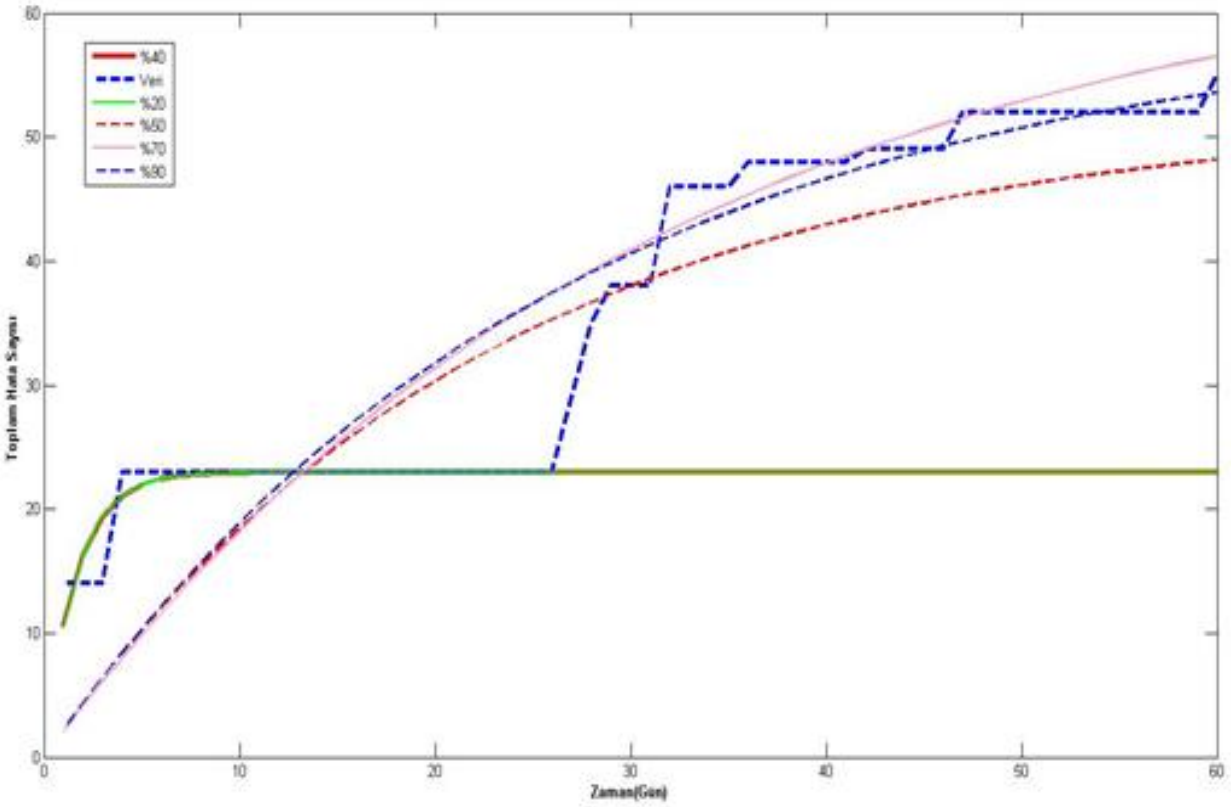
Şekil 22 Proje C Ortalama Hata Değerleri



	PNZ IFD	Geciktirilmiş S Şekilli	Musa Üstel	Büklümlü S Şekilli	Goel Okumoto	Yamada
Sum of MSE%70	70,8339	88,7253	86,6371	86,9936	54,7072	52,7642
Sum of MSE%30	2,5197	1,5	1,6	1,9	5,9	10,674
Sum of MSE2%50	80,5975	92,5668	67,6621	112,347	62,1251	65,5807
Sum of MSE%50	133,9964	98,245	95,043	85,576	28,8866	8,7466

Şekil 23 Proje D Ortalama Hata Değerleri

Şekil 20 Proje A Ortalama Hata Değerleri - Şekil 23 Proje D Ortalama Hata Değerleri'de A, B, C ve D projeleri için tüm modellerin MSE sonuçları grafikle beraber verilmiştir. Grafikte MSE%70 ve MSE2%50 parametre tahmini için kullanılan verinin modellenmesindeki MSE'yi belirtirken, MSE%30 ve MSE%50 ise kalan ve test için kullanılan verinin MSE değerlerini belirtmektedir.



Şekil 24 Proje D Farklı Yüzdelerde Modelleme

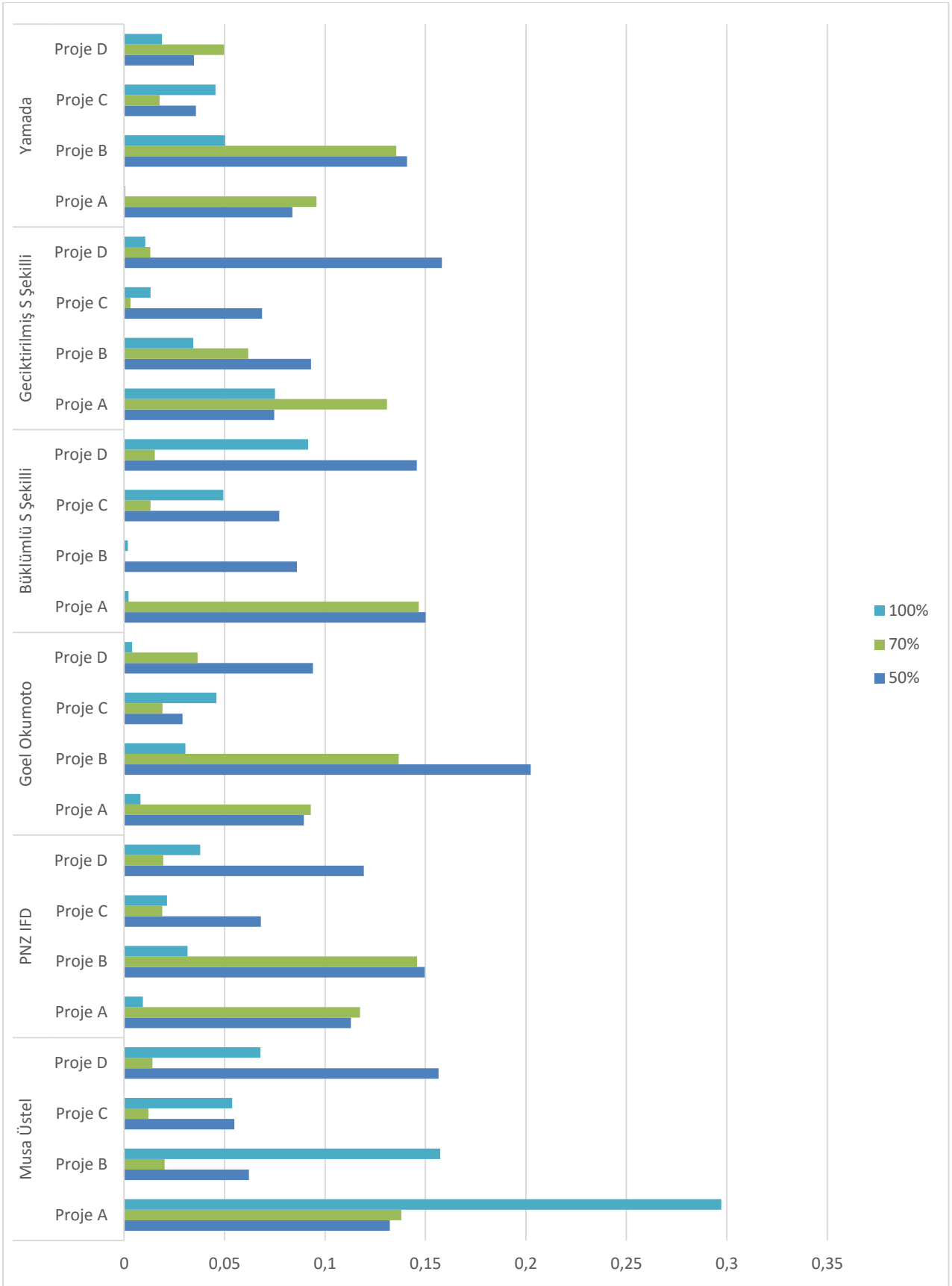
Şekil 24 Proje D Farklı Yüzdelerde Modelleme D projesinin Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modeli ile farklı yüzdelerde verinin kullanılması ile modellenmesinin grafikleri verilmiştir. Şekil 22 ve Şekil 23 incelendiğinde, C ve D projelerinde modellerin parametre tahmin etmekte kullanılan veriler üzerinde hesaplanan MSE değerlerinin, kalan veriler için hesaplanan MSE değerlerine göre yüksek olduğu görülmektedir. Normal koşullar altında parametre tahmin etmek için kullanılan veriler üzerinde hesaplanan MSE değerlerinin daha küçük olması beklenir. Bundan dolayı farklı yüzdelerde veri kullanılarak Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modeli için modelleme sonuçları hazırlanmış ve grafikte verilmiştir. Ayrıca her bir durum için MSE değerleri hesaplanmış ve Çizelge 13'te verilmiştir.

Çizelge 13 Farklı Durumlar için MSE Değerleri

Durum	MSE	MSE2
%20	459,0206	4,2604
%40	612,7218	2,1405
%50	28,8867	62,1937
%70	5,9	55,414
%90	1,0102	43,2228

Çizelge 13 incelendiğinde kullanılan veri yüzdelerinin artması ile MSE değerlerinin azaldığı görülebilir. Aynı zamanda projenin belirli bir süre sonunda olgunluğa ulaştığı ve belirli bir davranış gösterdiği görülebilir. Projenin ilk evrelerinde ise hata oluşumlarının belirli bir davranışının olmadığı bu nedenle modellemenin düşük başarımlı olduğu söylenebilir. Özellikle %40 ve %50 durumları incelendiğinde projedeki hata oluşumunda görülen ani değişimin modelleme sonucunu da etkilediği anlaşılmaktadır. Bu değişimden sonra yapılan modellemelerin ise genel olarak diğer durumlara göre başarılı olduğu görülebilir. Böylece model başarımında proje olgunluk seviyesinin önemli olduğu kararına varılabilir. Ayrıca modellerin veriye uyum sağlama konusunda özellikle kullanılan verilerin son değerine yakınsadığını görebiliriz. Bu durum parametre tahmininde kullanılan veri için hesaplanan MSE değerlerinin yüksek olmasına neden olmuştur.

Şekil 25 Projelerin A.BPRE Değerleri'te tüm modellerin dört proje ve üç durumu için A.BPRE değerlerinin mutlak değerlerinin grafiği toplamda en az değer alan modele göre sırası ile verilmiştir. A.BPRE değerlerinin hesaplanmasında, yazılım güvenilirlik modellerinin parametre tahmininde hata kayıt verilerinin %100'ü kullanıldığında A.BPRE değeri için de hata kayıt verilerinin %100'ü, %70'i kullanıldığında kalan %30 hata kayıt verisi, %50'si kullanıldığında ise kalan %50 hata kayıt verisi kullanılmıştır. Grafik incelendiğinde ise Yamada modelinin A.BPRE değerleri açısından genelde en iyi sonuç verdiği görülmüştür.



Şekil 25 Projelerin A.BPRE Değerleri

5. SONUÇLARIN DEĞERLENDİRİLMESİ ve GELECEK ÇALIŞMALAR

Bu çalışma kapsamında ALTAIR Savunma ve Yazılım Teknolojileri A.Ş. firmasının dört ayrı projesine ait ve faz bilgisinin kaydedilmemiş olduğu hata kayıt verileri kullanılarak yazılım güvenilirlik modellerinin uygunluğu test edilmiş ve karşılaştırılması yapılmıştır. Bu amaçla her bir proje için toplanan hata kayıt verilerinden kümülatif hata verileri elde edilmiş ve bu veriler seçilen yazılım güvenilirlik modelleri ile modellenmiştir. Kullanılacak yazılım güvenilirlik modellerinin seçiminde elde bulunan verilere uygun modellerin kullanılmasına özen gösterilmiştir. Bu amaçla endüstride ve araştırmalarda sıklıkla kullanılan homojen olmayan poisson süreci modellerinin kullanılmasına karar verilmiştir. Homojen Olmayan Poisson Süreci modelleri içerisinden birbirinden farklı dağılıma sahip altı ayrı model seçilmiş ve bu çalışma kapsamında değerlendirilmiştir. Homojen Olmayan Poisson Süreci yazılım güvenilirlik modelleri kolay uygulanabilir olmaları, analitik sonuçlar vermeleri ve sonuçlarının karşılaştırılabilir olması gibi özellikleri nedeniyle endüstride ve araştırmalarda sıklıkla kullanılan yazılım güvenilirlik modelleridir. Her bir Homojen Olmayan Poisson Süreci modelinin kendine özgü ortalama değer fonksiyonu vardır. Bu fonksiyonlarda bulunan parametreler sayesinde modeller farklı projelere uygulanabilmektedir. Her bir proje için ayrı ayrı yapılması gereken parametre tahminlerinin projeye uygun olması model başarımı için dikkat edilmesi gereken bir konudur. Bu çalışmada modellerin parametrelerinin tahmin edilmesinde en çok olabilirlik tahmin yöntemi kullanılmıştır ve her bir proje ve model için kümülatif hata verilerinden parametreler tahmin edilmiştir. En çok olabilirlik tahmin yöntemi, yalnızca yazılım güvenilirlik modellerinde değil, tüm modelleme alanlarında parametre tahmininde kullanılan bir yöntemdir. Parametreleri tahmin edilen modellerin ortalama değer fonksiyonları elde edilmiş ve bu fonksiyonlar gerçek kümülatif hata verileri ile karşılaştırılmıştır. Bu çalışma kapsamında verilerin %100'ünün, ilk %70'inin ve ilk %50'sinin parametre tahmininde kullanılması durumları olmak üzere üç ayrı durum incelenmiştir. İlk olarak her bir proje kümülatif hata verisinin tamamı kullanılarak parametre tahmini yapılmış ve yine verinin tamamı ile model performansı değerlendirilmiştir. Parametre tahmininde verilerin %70 ve %50'sinin kullanıldığı durumlarda ise, model performansını değerlendirmek için verilerin sırası ile kalan %30 ve %50'si kullanılmıştır. Model

performanslarının ölçülmesi için hata kareler ortalaması, ortalama bağıl hata ve PRED ölçümleri yapılmıştır. Parametre tahmininde verilerin %100, %70 ve %50'sinin kullanıldığı üç durumda da her bir model ve proje için MSE, MMRE, PRED ve A.BPRE ölçümleri alınmış ve sonuçlar kaydedilmiştir. Sonuçlar ayrı ayrı değerlendirildiğinde parametre tahmininde verilerin %100'ünün kullanıldığı durumda MSE değerleri bakımından A ve D projelerinde Yamada modelinin, B ve C projelerinde ise Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci modelinin en iyi sonucu verdiği görülmektedir. Aynı durumda MMRE değerleri bakımından A projesinde Üstel Homojen Olmayan Poisson Süreci Goel Okumoto, B projesinde Büklümlü S Şekilleri modeli, C projesinde Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci modeli, D projesinde ise Yamada modeli en iyi sonucu vermiştir. PRED değerleri bakımından ise Üstel Homojen Olmayan Poisson Süreci Goel Okumoto A projesinde, Büklümlü S Şekilleri modeli B ve C projelerinde, Yamada modeli ise D projesinde en iyi sonucu vermiştir. A.BPRE değerleri bakımında A projesinde Yamada modeli, B projesinde Büklümlü S Şekilleri modeli, C projesinde Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci modeli ve D projesinde ise Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modeli en iyi sonucu vermiştir. Parametre tahmininde verilerin ilk %70'inin kullanıldığı durumda MSE, MMRE, PRED ve A.BPRE değerleri bakımından A projesinde Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modeli, B projesinde Büklümlü S Şekilleri modeli, C ve D projesinde ise Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci modeli en iyi sonuçları vermişlerdir. Parametre tahmininde verilerin ilk %50'sinin kullanıldığı durumda MSE, MMRE, PRED ve A.BPRE değerleri bakımından D projesinde Yamada modeli, B projesinde Musa Üstel modeli, C projesinde ise Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modeli en iyi sonuçları vermişlerdir. A projesinde ise MSE, MMRE ve PRED değerleri bakımından en iyi sonucu Yamada modeli verir iken A.BPRE değeri bakımında ise en iyi sonucu Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci modeli vermektedir. Genel olarak sonuçlar incelendiğinde her bir proje ve durum için farklı modellerin en iyi sonucu verebildikleri görülmektedir. Bu çalışma kapsamında yapılan model performans ölçümlerinden direk olarak bir modelin diğerlerine göre daha iyi modelleme yaptığı söylenememektedir. Bu nedenle parametre tahmininde kullanılan veri yüzdesi durumları da dikkate alınarak her bir durum ve proje için MSE, MMRE, PRED ve A.BPRE ölçümleri bakımından en iyi sonuç veren modeller

belirlenmiş ve Çizelge 14 En İyi Değer Alan Sonuçların Değerlendirilmesi'te verilmiştir.

Çizelge 14 En İyi Değer Alan Sonuçların Değerlendirilmesi

Modeller	MSE	MMRE	PRED	A.BPRE	Toplam Durum Sayısı
Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci	$B_{\%100}$, $C_{\%100}$, $C_{\%70}$, $D_{\%70}$	$C_{\%100}$, $C_{\%70}$, $D_{\%70}$	$C_{\%70}$, $D_{\%70}$	$C_{\%100}$, $C_{\%70}$, $D_{\%70}$, $A_{\%50}$	13
Yamada	$A_{\%100}$, $A_{\%50}$, $D_{\%100}$, $D_{\%50}$	$A_{\%50}$, $D_{\%100}$, $D_{\%50}$	$A_{\%50}$, $D_{\%100}$, $D_{\%50}$	$A_{\%100}$, $D_{\%50}$	12
Üstel Homojen Olmayan Poisson Süreci Goel Okumoto	$A_{\%70}$, $C_{\%50}$	$A_{\%100}$, $A_{\%70}$, $C_{\%50}$	$A_{\%100}$, $A_{\%70}$, $C_{\%50}$	$D_{\%100}$, $A_{\%70}$, $C_{\%50}$	11
Büklümlü S Şekilleri	$B_{\%70}$	$B_{\%100}$, $B_{\%70}$	$B_{\%100}$, $B_{\%70}$, $C_{\%100}$	$B_{\%100}$, $B_{\%70}$	8
Musa Üstel	$B_{\%50}$	$B_{\%50}$	$B_{\%50}$	$B_{\%50}$	4
PNZ IFD	-	-	-	-	0

Çizelge 15 En İyi İkinci Değer Alan Sonuçların Değerlendirilmesi

Modeller	MSE	MMRE	PRED	A.BPRE	Toplam Durum Sayısı
Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci	$B_{\%50}$	$B_{\%50}$, $A_{\%50}$	$B_{\%50}$	$D_{\%100}$	5
Yamada	$A_{\%70}$, $C_{\%50}$	$A_{\%70}$, $C_{\%50}$	$A_{\%70}$, $C_{\%50}$	$A_{\%70}$, $C_{\%50}$, $A_{\%50}$	9
Üstel Homojen Olmayan Poisson Süreci Goel Okumoto	$A_{\%100}$, $D_{\%100}$, $A_{\%50}$, $D_{\%50}$	$B_{\%100}$, $D_{\%100}$, $D_{\%50}$	$B_{\%100}$, $D_{\%100}$, $D_{\%50}$, $A_{\%50}$	$B_{\%100}$, $D_{\%50}$	13
Büklümlü S Şekilleri	$C_{\%100}$	$C_{\%100}$	$C_{\%100}$	$A_{\%100}$, $B_{\%50}$	5
Musa Üstel	$B_{\%70}$, $C_{\%70}$, $D_{\%70}$	$B_{\%70}$, $C_{\%70}$, $D_{\%70}$	$B_{\%70}$, $C_{\%70}$, $D_{\%70}$	$B_{\%70}$, $C_{\%70}$, $D_{\%70}$	12
PNZ IFD	$B_{\%100}$	$A_{\%100}$	$A_{\%100}$	$C_{\%100}$	4

A_i = A projesi $i.$ durum

B_i = B projesi $i.$ durum

C_i = C projesi $i.$ durum

D_i = D projesi $i.$ durum

%100; Proje verilerinin tamamının kullanıldığı durum

%70; Proje verilerinin %70'inin kullanıldığı durum

%50; Proje verilerinin %50'sinin kullanıldığı durum

Çizelge 14'ten de anlaşılacağı üzere bu çalışma kapsamında her bir yazılım güvenilirlik modeli dört ayrı proje, üç ayrı durum ve yine dört ayrı ölçüm yöntemi kullanılarak toplamda kırk sekiz durum için değerlendirilmiştir. Bu kırk sekiz durum içerisinden her bir durumda hangi modelin en iyi sonucu verdiği belirlenmiştir. Çizelge 14 En İyi Değer Alan Sonuçların Değerlendirilmesi'te de görülebildiği gibi Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci toplamda on üç durum için en iyi model olmuştur. Daha sonra sırasıyla Yamada modeli ve Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modelleri on iki ve on bir durumda en iyi sonucu veren modeller olarak ikinci ve üçüncü model olmuşlardır. Toplam durumlar incelendiğinde sonuçların birbirine yakın olması nedeni ile Geciktirilmiş S Şekilli Homojen Olmayan Poisson Süreci modelinin diğer modellere göre daha iyi olduğu söylenemez. Bununla birlikte özellikle Yamada ve Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modellerinin yapılan MSE, MMRE, PRED ve A.BPRE ölçümleri bakımından en iyi sonuç verdiği durumlar incelendiğinde bu iki modelin diğer modellerden farklı olarak birbirlerine yakın sonuçlar verdikleri görülebilmektedir. Yapılan performans ölçümleri incelendiğinde bu modellerden birinin birinci olduğu durumlarda diğer modelin hep ikinci olduğu görülebilir. İki model incelendiğinde iki modelin birbirinden hata miktarı fonksiyonları ($a(t)$) nedeniyle ayrıldıkları görülebilir. Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modelinin hata miktarı fonksiyonu $a(t) = a$ iken bu fonksiyon Yamada modeli için $a(t) = ae^{\alpha t}$ dir. Bu çalışma kapsamında yapılan parametre tahminleri incelendiğinde Yamada modelinin α parametrelerinin hep çok küçük değerler olarak tahmin

edildiği görülebilir. Bu sonuç bize bu projelerde yazılıma zamanla yeni hata eklenmesinin az miktarda olduğunu göstermektedir. Ayrıca bu durum Yamada modeli ve Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modelinin birbirine benzer sonuçlar vermesine neden olmuştur. Çizelge 14 ve 15 incelendiğinde ise Yamada modeli ve Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modelinin birbirlerini takip eden sonuçlar verdiği görülmektedir. Bu yaklaşımla iki modelin birbirine yakınsadığı düşünüldüğünde toplamda yirmi üç durum için en iyi sonucu bu modellerin vermiş olduğu çıkarılabilir ve bu sonuçlar ile Yamada ve Üstel Homojen Olmayan Poisson Süreci Goel Okumoto modellerinin web yazılımlarını modellemede en iyi modellemeyi yapacakları kararına varılabilir. Böylelikle giriş bölümünde verilen ikinci araştırma sorusu olan Homojen Olmayan Poisson Süreci yazılım güvenilirlik modellerinin web yazılımlarındaki performansları ve birbirlerine kıyasla nasıl başarı sağladıkları sorusunun cevabı verilmiştir.

Bu çalışma kapsamında ele alınan web yazılımlarının hata kayıtları incelendiğinde hata oluşumlarının rassal olduğu görülebilir. Aynı zamanda hata oluşumlarının belirli bir periyodu takip etmediği de anlaşılabilir. Bu özellikleri dikkate alındığında eldeki web yazılımlarındaki hata oluşumlarının Homojen Olmayan Poisson Süreci modellerinin varsayımlarını karşıladıklarını düşünülmektedir. Bu nedenle web yazılımlarının güvenilirliklerinin modellenmesinde Homojen Olmayan Poisson Süreci güvenilirlik modellerinin kullanılabileceği kararına varılmıştır. Böylelikle giriş bölümünde verilen ilk araştırma sorusu olan Web yazılımlarında Homojen Olmayan Poisson Süreci yazılım güvenilirlik modelleri kullanılabilirliği sorusunun cevabı verilmiştir.

Yapılan çalışmanın sonucunda hata kayıtlarının %70 ve %50'sindeki veriler kullanılarak tahmin edilen parametreler oluşturulan modeller için elde edilen başarım ölçümleri incelendiğinde özellikle biten iki proje için tüm modellerin yüksek başarımla tahminleme yaptığı, fakat aynı başarının diğer projelerde sağlanamadığı görülebilir. Bu durum nedeniyle modellemenin başarımını projenin olgunluk seviyesinin gösterdiği söylenebilir. Başka bir deyişle bir projede yazılım güvenilirliğinin modellenmesi sırasında projenin beklenen bitim süresinin yarısına gelindikten sonra yapılacak modellemelerin başarılı sonuçlar verebileceği kararına varılabilir. Bu şekilde kayıtların %70 ve %50'sindeki veriler kullanılarak oluşturulan modellerin hataların tahmin edilmesinde ne kadar başarılı olacağı açıklanmıştır.

Bir çalışmanın geçerliliği, çalışma sonuçlarına olan güveni belirler ve sonuçların araştırmacının yorumlaması nedeniyle taraflı olmamasını gerektirir. Çalışmanın geçerliliği çalışmaya başlamadan önce düşünülmesi gereken bir konudur [41].

Geçerliliği etkileyebilecek farklı faktörler vardır. Aşağıda bu faktörleri ve yapılan çalışmanın bu faktörler açısından değerlendirilmesi anlatılmıştır.

- Güvenilirlik: Bir çalışmanın güvenilir olması o çalışmanın araştırmacıdan araştırmacıya farklı sonuç vermemesi ile alakalıdır. Başka bir araştırmacı aynı verilerle çalışmayı tekrarladığında yine aynı sonuca ulaşmalıdır [41]. Bu konu toplanan verilerin nasıl toplandığının ve nasıl kullanıldığının açık bir şekilde anlatılması ile giderilir. Bu çalışmada kullanılan tüm veriler açık bir şekilde verilmiştir ve yapılan işlemler sırası ve ara sonuçları ile anlatılmıştır. Bu bakımdan bu çalışmaya güvenilir denilebilir.
- İç tehditler: İç tehditler çalışmayı etkileyebilecek faktörlerin ne kadar ele alındığı ile ilgilidir [41]. Çalışmayı etkileyebilecek faktörler belirlenmeli ve bu faktörlerin çalışmaya etkisi incelenmelidir. Bu çalışmada modellerin karşılaştırılması sırasında farklı projelerin farklı sonuçlara neden olabileceği düşünülmüş ve çalışma dört ayrı proje ile yapılmıştır. Çalışma kapsamına alınacak projelerin seçiminde projelerin bitip bitmeme durumu, yaşam döngüleri, geliştirilme ortamları dikkate alınmış bu sayede bu faktörlerin sonuca etkileri azaltılmaya çalışılmıştır. Kullanılan dört projenin ikisi bitmiş proje iken diğer ikisi henüz bitmemiş projelerden seçilmiştir. Aynı zamanda iki proje şelale yaşam döngüsüne sahipken diğer iki proje spiral yaşam döngüsüne sahiptir. İç tehdit olabilecek bir faktör yazılım geliştirme ortamıdır. Çalışma kapsamına alınan tüm projeler Java yazılım dili ile Windows Eclipse ortamında geliştirilmiştir bu da bu çalışmanın yazılım dili ve geliştirme ortamının sonuca etkilerini belirlemede yetersiz kalmasına neden olmuştur. Bununla birlikte Java ve Windows Eclipse web yazılımlarında sıklıkla kullanılmaktadır. Bir diğer iç tehdit olabilecek durum ise parametre tahminlerinin yapılmasında en çok olabilirlik yönteminin kullanılmasıdır. Tahmin yöntemi olması dolayısıyla parametre tahminlerini ve model performansını direk etkilemektedir, ancak yapılan çoğu çalışmada model parametrelerinin tahmin edilmesinde sıklıkla kullanılan bir yöntemdir. Ayrıca yazılım güvenilirlik modellerinin performans ölçümünde bir iç tehdit

oluşturabilir. Bu nedenle MSE, MMRE, PRED ve A.A.BPRE hesaplamaları birlikte kullanılmış ve bu iç tehdidin etkisi azaltılmaya çalışılmıştır. Bu çalışmada hata kayıt verilerin kullanılan projelerin tamamının aynı şirkete ait olması iç tehdit olarak düşünülebilir. Projede çalışan personeller, şirket içi geliştirme süreçleri, müşteriler gibi birçok faktör yazılım hata oluşumunu ve dolayısı ile yazılım güvenilirliğini etkileyebilir. Bu çalışma kapsamına alınan tüm projelerin aynı şirkete ait olması çalışma sonuçlarının sadece bu şirkete özel olmasına neden olabilir.

- Genelleme: Genelleme bir çalışmada elde edilen sonuçların benzer bir durumda kullanılıp kullanılamayacağı ile ilgilidir [19]. Bu çalışma kapsamında elde edilen sonuçlarda yalnızca en iyi yazılım güvenilirlik modeli seçilmemiştir. Ayrıca bu modellerin performansları sıralanmıştır. Bu nedenle benzer bir çalışmada yol gösterici olabileceği düşünülmektedir. Fakat günümüzde geliştirilmiş olan çok sayıda yazılım güvenilirlik modeli vardır ve bu çalışmada sadece altı modelin performansı incelenmiştir. Bu durum çalışmanın genellemesine engel bir faktör olabilir.

Bu çalışma kapsamında incelenen projelerin hepsi ALTAIR Savunma ve Yazılım Teknolojileri A.Ş. firmaya ait projelerdir. Firmaya has yazılım geliştirme teknikleri veya süreçleri çalışma sonucunu etkileyebileceği düşünülmektedir. Bu nedenle benzer bir çalışmanın farklı firmaların projeleri için de tekrarlanmasında fayda görülmektedir. Aynı zamanda yapılacak ileriki çalışmalarda zaman serisi analizi ya da farklı yazılım güvenilirlik modellerinin uygulanması denenmelidir. Ayrıca projelerin hata kayıt verilerinin farklı bölümlerinin farklı yazılım güvenilirlik modelleri ile modellenebilir. Yapılacak çalışmalarda web yazılımlarının yanı sıra farklı tür yazılımlar için de güvenilirlik analizlerinin yapılması güvenilirlik analizlerinde web yazılımlarının diğer yazılımlara göre içerdiği farklılıkları belirlemede faydalı olabilir. Aynı zamanda farklı tür yazılımların güvenilirliklerinin modellenmesi için ne tür yazılım güvenilirlik modellerinin kullanılabileceği ile ilgili çalışmalar yapılabilir. Bunun yanında yapılacak yeni çalışmalarda bu çalışmanın örnek teşkil edebileceği düşünülmektedir.

KAYNAKLAR

[1] Kurtel, K. ve Eren, Ş., "Quality Requirements for Software Architecture: Software Reliability," *Journal of Computer Science and Engineering,* vol. 4, pp. 75-83, 2011.

[2] Group, S. "Standish group chaos," 2014. Available: http://www.standishgroup.com. [Accessed Kasım 2015]. [Online].

[3] Quyoum, A., Dar, M. U. D. ve Quadri, S. M. K., "Improving Software Reliability using Software Engineering Approach- A Review," *International Journal of Computer Applications,* vol. 10(5), pp. 41-47, 2010.

[4] Türkan, A. H., "Güvenilirlik Analizinde Kullanılan İstatistiksel Dağılım Modelleri," Yüksek Lisans, İstatistik Anabilim Dalı, Çukurova Üniversitesi, 2007.

[5] Carmines, E. G. ve Zeller, R. A., *Reliability and Validity Assessment*, 9 ed.: Sara MillerMcCune, Sage Publications, Inc, 1979.

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

[7] Goel, A. L., "Software Reliability Models: Assumptions, Limitations, and Applicability," *IEEE Transactions on Software Engineering,* vol. 11(12), pp. 1411-1423, 1985.

[8] Sk.Md.Rafi, Rao, D. K. N., Sety, D. S. P. ve Akthar, S., "Two types of Imperfect Debugging Software Reliability Growth model with Warranty Cost and Release Time Determination," *International Journal of Software Engineering Research & Practices,* vol. 2(4), pp. 1-8, 2012.

[9] Aydın, A., "İteratif Yazılım Geliştirme için Hata Tahminleme Modeli Araştırması: Bir Durum Çalışması," Yüksek Lisans, Bilgisayar Mühendisliği, Hacettepe Üniversitesi, 2014.

[10] Kaur, K. ve Anand, S., "Review on Software and Hardware Reliability and Metrics," *International Journal of Science, Engineering and Technology Research,* vol. 2(5), pp. 1108-1110, 2013.

[11] Rosenberg, D. L. ve Brennan, D., "Application and Improvement of Software Reliability Models," 2001.

[12] Pham, H., "Software Reliability and Cost Models Perspectives, Comparison, and Practice," *European Journal of Operational Research,* vol. 149, pp. 475-489, 2002.

[13] Hoff, A., "Software Reliability," Avaliable http://www.uwplatt.edu/files/csse/courses/prev/csse411-materials/s10/SoftwareReliability_hoffaa.docx [Accessed Kasım 2015]. [Online].

[14] Offutt, J., "Web Software Applications Quality Attributes," *Quality Engineering in Software Technology,* vol. 11, pp. 187-198, 2002.

[15] Mendes, E., "Web Development Versus Software Development," in *A Pathway to Improve Software Effort Estimation*, ed: Springer Berlin Heidelberg, 2014, pp. 13-25.

[16] Deshpande, Y., Murugesan, S., Ginige, A., Hansen, S., Schwabe, D., Gaedke, M.*, et al.*, "Web Engineering," *Journal of Web Engineering,* vol. 1 (1), pp. 3-17, 2002.

[17] Elsanhouri, A. E., Ahmed, M. A. ve Abdullah, A. H., "Cloud Applications Versus Web Applications: A Differential Study," presented at the The First International Conference on Communications, Computation, Networks and Technologies, Venice, Italy, 2012.

[18] Offutt, J., "Web Software Applications Quality Attributes," presented at the Quality Engineering in Software Technology, Nuremberg, Germany, 2002.

[19] Rana, R., Staron, M., Berger, C., Hansson, J., Nilsson, M., Törner, F.*, et al.*, "Selecting Software Reliability Growth Models and Improving Their Predictive Accuracy Using Historical Projects Data," *Journal of Systems and Software,* vol. 98, pp. 59-78, 2014.

[20] Lai, R. ve Garg, M., "A Detailed Study of NHPP Software Reliability Models (Invited Paper)," *Journal of Software,* vol. 7, 2012.

[21] Yamada, S., *Software Reliability Modelling Fundamentals and Applications*. Tokyo: Springer, 2014.

[22] Pham, H., *System Software Reliabililty* London: Springer, 2006.

[23] Qutaish, R. E. A. ve Abran, A., "An Analysis of the Design and Definitions of Halstead's Metrics," presented at the Proceeding of 15th International Workshop on Software Measurement, Monttreal Ouebec, Canada, 2005.

[24] Mohd, R. ve Nazir, M., "Software Reliability Growth Models: Overview and Applications," *Journal of Emerging Trends in Computing and Information Sciences,* vol. 3 (9), pp. 1309-1320, 2012.

[25] Liu, J., "Function Based Nonlinear Least Squares and Application to Jelinski–Moranda Software Reliability Model," *arXiv pre print arXiv: 1108.5185,* vol. 1, pp. 1-26, 2011.

[26] Ho, S. L., Xie, M., ve Goh, T. N., "A Study of the Connectionist Models," *Compuers and Mathematics with Applications,* vol. 46, pp. 1037-1045, 2003.

[27] Pham, H., "An imperfect-debugging fault-detection dependent-parameter software," *International Journal of Automation and Computing,* vol. 4, pp. 325-328, 2007.

[28] Xiea, M., Hongb, G. Y. ve Wohlinc, C., "Modeling and Analysis of Software System Reliability," in *Case Studies in Reliability and Maintance* W. R. B. a. D. N. P. Murthy, Ed., ed: Wiley, Newyork, 2003, pp. 233-249.

[29] Nagar, P. ve Thankachan, B., "Application of Goel-Okumoto Model in Software Reliability Measurement," *Special Issue of International Journal of Computer Applications,* vol. 11, pp. 1-3, 2012.

[30] Hong, G. Y., Xie, M., M.Zhao ve Wohlin, C., "Interval Estimation in Sofware Reliability Analysis," presented at the Proceedings 4.th International Applied Statistics in Industry Conference, Kansas City, Missouri, US, 1997.

[31] Williams, D. R. P., "Prediction Capability Analysis of Two and Three Parameters Software Reliability Growth Models," *Information Technology Journal,* vol. 5(6), pp. 1048-1052, 2006.

[32] Ahmada, N., Khanb, M. G. M. ve Rafib, L. S., "Analysis of an Inflection S-shaped Software Reliability Model Considering Log-logistic Testing-Effort and Imperfect Debugging," *International Journal of Computer Science and Network Security,* vol. 11(1), pp. 161-171, 2011.

[33] Zhoa, M. ve Xiea, M., "On Maximum Likelihood Estimation for a General Non-homogeneous Poisson Process," *Board of the Foundation of the Scandinavian Journal of Statistics,* vol. 23, pp. 597-607, 1996.

[34] Kapur, P. K., Pham, H., Gupta, A. ve Jha, P. C., *Software Reliability Assessment with OR Application* Springer, 2011.

[35] Ullah, N., Morisio, M. ve Vetro, A., "A Comparative Analysis of Software Reliability Growth Models using defects data of Closed and Open Source Software," presented at the IEEE 35th Software Engineering Workshop, Torino, Italy, 2012.

[36] Downs, T. ve Scott, A., "Evaluating the Performance of Software Reliability Models," *IEEE Transactions on Reliability,* vol. 41 (4), pp. 533-538, 1992.

[37] Chang, Y.-P., "Estimation of Parameters For Nonhomogeneous Poissonprocess: Software Reliability With Change-Point Model," *Communications in Statistics - Simulation and Computation,* vol. 30(3), pp. 623-635, 2001.

[38] Tabak, B. A. ve Demirörs, O., "Efor Kestirim Doğruluğu İçin Tasarım Büyüklüğü ve Problem Büyüklüğü Karşılaştırılması," presented at the Turkish National Software Engineering Symposium, İzmir, Turkey, 2013.

[39] Chouseinoglou, O. ve Aydın, Ö. M., "A Fuzzy Model of Software Project Effort Estimation," *An Official Journal of Turkish Fuzzy Systems Association,* vol. 4, pp. 68-76, 2013.

[40] Port, D., Korte, M., ve Idquo, "Comparative Studies of the Model Evaluation Criterions MMRE and PRED in Software Cost Estimation Research, &rdquo, Proc. Second ACM-IEEE Int', I Symp. Emprical Software Engineering and Measurement, pp.51-60, 2008."

[41] Runeson, P. H., Martin, "Guidelines for Conducting and Reporting Case Study Research in Software Engineering," *Empirical Software Engineering,* vol. 14, pp. 131-164, 2008.

ÖZGEÇMİŞ

Kimlik Bilgileri

Adı Soyadı : Rabia Burcu Karaömer
Doğum Yeri : Pozantı
Medeni Hali : Evli
E-posta : r_burcuglbhr@hotmail.com
Adresi : Serhat mh. 1297. Cad No:6 Nevbahçe 1 Konutları B Blok Daire No:14 Yenimahalle ANKARA

Eğitim

Lisans : İstatistik Bölümü, Hacettepe Üniversitesi (Ocak 2011)
Yüksek Lisans : Endüstri Mühendisliği, Hacettepe Üniversitesi (Devam)

Yabancı Dil ve Düzeyi

İngilizce : Çok iyi

İş Deneyimi

07/2014- : Süreç Mühendisi, Proven Bilişim Teknolojileri, Ankara
12/2012-07/2014 :Sürüm Yönetimi Uzmanı, Korvus Bilişim Teknolojileri Ltd. Şti., Ankara
04/2012-12/2012 : Proje Uzmanı, Milsoft Yazılım Teknolojileri A.Ş., Ankara

Deneyim Alanları

Tezden Üretilmiş Projeler ve Bütçesi

Tezden Üretilmiş Yayınlar

Tezden Üretilmiş Tebliğ ve/veya Poster Sunumu ile Katıldığı Toplantılar

---