BAŞKENT ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YAZILIM MÜHENDİSLİĞİ PROJELERİNDE HATA ÖLÇÜMÜ, KÖK NEDEN ANALİZLERİ VE ÖRNEK BİR VAKA İNCELEMESİ

ÇAĞLA ATAGÖREN

YÜKSEK LİSANS TEZİ

2012

YAZILIM MÜHENDİSLİĞİ PROJELERİNDE HATA ÖLÇÜMÜ, KÖK NEDEN ANALİZLERİ VE ÖRNEK BİR VAKA İNCELEMESİ

DEFECT MEASUREMENT IN SOFTWARE PROJECTS, ROOT CAUSE ANALYSIS AND A CASE STUDY

ÇAĞLA ATAGÖREN

Başkent Üniversitesi

Lisansüstü Eğitim Öğretim ve Sına v Yönetmeliğinin

İSTATİSTİK ve BİLGİSAYAR Bilimleri Anabilim Dalı İçin Öngördüğü

YÜKSEK LİSANS TEZİ

olarak hazırlanmıştır.

2012

“YAZILIM MÜHENDİSLİĞİ PROJELERİNDE HATA ÖLÇÜMÜ, KÖK NEDEN ANALİZLERİ VE ÖRNEK BİR VAKA İNCELEMESİ” başlıklı bu çalışma, jürimiz tarafından, 12/09/2012 tarihinde**, İSTATİSTİK ve BİLGİSAYAR BİLİMLERİ ANABİLİM DALI'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Başkan

Prof. Dr. İsmail ERDEM

Üye (Danışman)

Doç. Dr. Mehtap AKÇİL

Üye

Prof. Dr. Ali HALICI

ONAY

…./09/2012

Prof. Dr. Emin AKATA
Fen Bilimleri Enstitüsü Müdürü

TEŞEKKÜR

Yüksek Lisans eğitimimin bu tez ile sonuna gelmiş bulunuyorum. Bu çalışma ile mesleğimde de yol kat ederek, kendime yeni bakış açıları kazandırabildim. İleride bu şekilde çok farklı ilerlemeler ve bakış açıları kazanacağıma inanarak,

Yardımlarından dolayı *Doç. Dr. Mehtap AKÇİL*'e,

Lisans eğitimim boyunca ilminden faydalandığım, insani ve ahlaki değerleri ile de örnek aldığım, onunla çalışmaktan onur duyduğum ve ayrıca tecrübelerinden yararlanırken göstermiş olduğu hoşgörü ve sabrından dolayı değerli hocam, *Öğretim Görevlisi Umut HÜSEYİNOĞLU*'na,

Araştırmanın analizi sırasında yardımlarını eksik etmeyen *şirket çalışanlarına,*

Çalışmamın ilk gününden son gününe kadar destek olan *ailem* ve *Emrah TÜZÜN'e* teşekkürü bir borç biliyorum.

ANKARA, EYLÜL, 2012

Çağla ATAGÖREN

ÖZ

YAZILIM MÜHENDİSLİĞİ PROJELERİNDE HATA ÖLÇÜMÜ, KÖK NEDEN ANALİZLERİ VE ÖRNEK BİR VAKA İNCELEMESİ

ÇAĞLA ATAGÖREN
Başkent Üniversitesi Fen Bilimleri Enstitüsü
İstatistik ve Bilgisayar Bilimleri Anabilim Dalı

Yazılım projelerinde, müşteriye verilen son ürünün, müşteri isteklerine uygun olmasının yanında en az hata ile teslim edilmesi amaçlanmaktadır. Bu sebeple, müşteriye ürün teslim edilmeden önce düzeltilmek amacıyla bulunan hatalar süreç boyunca kayıt altına alınır. Böylece kayıt altına alınarak biriktirilen hatalar daha sonra incelenebilir.

Bu çalışmada yazılım projelerinin doğası ve içeriklerine ve hata kavramına ayrıntılı bir şekilde değinilmiştir. Süreç boyunca hataların azaltılması ve ortadan kaldırılması için uygulanan tekniklere yer verilmiştir. Türkiye'nin önde gelen orta ölçekli yazılım şirketlerinden birinde örnek bir vaka incelemesi yapılmıştır. Kök neden analizi yöntemi ve bu yöntemi destekleyen Pareto ve sebep-sonuç grafikleri yardımıyla incelenmiştir. İnceleme esnasında uygulanan tüm aşamalar ve örnek vaka incelemesi ayrıntılı bir biçimde anlatılmıştır.

ANAHTAR SÖZCÜKLER: yazılım projelerinde hata analizi, Pareto grafikleri, sebep-sonuç grafikleri, kök neden analizi, balık kılçığı diyagramları

Danışman: Doç. Dr. Mehtap Akçil, Başkent Üniversitesi, İstatistik ve Bilgisayar Bilimleri Bölümü

ABSTRACT

DEFECT MEASUREMENT IN SOFTWARE PROJECTS, ROOT CAUSE ANALYSIS AND A CASE STUDY

ÇAĞLA ATAGÖREN

Başkent University Institute of Science and Engineering,

Department of Statistics and Computer Science

In software projects, final products aim to meet customer needs and concurrently to have the least number of defects. Therefore, defects are recorded during the software development process with the intention of fixing them before the product is delivered to the customers.

In this study, the special structure of software projects and the concept of software defects are researched in detail. The techniques that are used for reducing or eliminating the defects throughout the process are presented. A case study is conducted in one of the leading, medium sized software companies of Turkey. The root cause analysis and supporting Pareto analysis along with cause-effect diagrams. All stages of the research and the case study are explained in detail.

KEY WORDS: defect measurement analysis in software projects, Pareto charts, cause-effect charts, root cause analysis, fish bone diagrams

Supervisor: Assoc. Prof. Mehtap Akçil, Başkent University, Statistics and Computer Science Department

İÇİNDEKİLER LİSTESİ

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

TABLOLAR DİZİNİ

Sayfa

KISALTMALAR

A	Araçlar
CMMI	Capability Maturity Model Integration (Yetenek Olgunluk Model Entegrasyonu)
DHS	Dikey Hata Sınıflandırma
DOK	Dokümantasyon
G	Girdiler
GER	Gereksinimler
GH	Grafiksel Hata
INFAI	Informal Action Item (Resmi Olmayan İşlem Maddesi)
K	Kişiler
KSS-SLOC	Kod Satır Sayısı (Source Lines of Code)
KLOC	Kilo Lines of Code (Kod Satır Sayısı /100)
M	Materyaller
MAN	Mantık
M/A	Metot/Araçlar
MR	Modification Request (Modifikasyon İsteği)
PR	Peer Review (Akran Değerlendirmesi)
REQ	Requirements (Gereksinimler)
RUP	Rational Unified Process (Rasyonel Birleştirilmiş Süreç)
TSRM	Tasarım
SYGM	Sosyal Yardımlar Genel Müdürlüğü
SCR	Software Change Request (Yazılım Değişiklik İstekleri)

1. GİRİŞ

Günümüz projelerinde süreç yönetimi gün geçtikçe önem kazanmaktadır. Sürecin yönetilmesiyle birlikte daha verimli projeler yapılmaya başlanmış, projelerin başarı oranları artmıştır. Yapılan projede, ister yazılım projesi ister üretim yapan bir fabrikanın projesi olsun, ortak amaç müşteriye teslim edilen son ürün üzerinde en az hatanın olmasıdır. Süreç iyileştirmelerinin bir etkisi olarak, projelerin başladığı günden müşteri teslimatı yapılan güne kadar geçen süreçte bulunan hatalar tespit edilir ve kayıt altına alınır. Kayıt altına alınan verinin kalitesi ve güvenilirliği yüksek olmalıdır. Aksi takdirde yapılacak olan analizde yanlış sonuçlar ve yorumlar ortaya çıkabilmektedir. Analizler sonucunda toplanan veriler doğru kullanılıp yorumlandığında ancak anlamlı sonuçlar vermektedir. Kötü yapılmış bir analiz, sonucu gerçekten uzaklaştırabilmektedir. Dolayısıyla analiz kapsamında kullanılan teknik ve yöntem doğru seçilmelidir. Giderek artan yazılım ihtiyacı büyük ve kritik projelerde ciddi bir süreç yönetimini zorunlu kılmaktadır. Aksi halde, karmaşık veya büyük projelerin başarıya ulaşması mümkün değildir.

Bu çalışmada amaç; ortaya çıkan yazılım hatalarının doğru bir biçimde değerlendirilmesi için kullanılabilecek yöntemleri sıralamak ve örnek bir vaka incelemesi ile hataların değerlendirilip kök sebeplerinin tespit edilmesini sağlamaktır. Bulunan kök sebepler ve bu sebepler için alınan düzeltici önlem planları sayesinde, süreçte yapılan hataların projelerin ilerleyen aşamalarında tekrarlanması önlenerek, hataların giderilmesi sağlanmaya çalışılmaktadır. Analiz kapsamında kök sebepler belirlendikten sonra yapılacak olan iyileştirme bu çalışma kapsamında yer almamaktadır. Fakat ilerleyen çalışmalarla bu iyileşme de takip edilebilir.

Yapılan tez çalışmasının bu bölümünde, araştırma hakkında genel bilgi verilmektedir.

Bölüm 2'de yazılım projeleri ve yapıları hakkında genel bilgi yazılım projeleri hakkında yapılmış iki farklı araştırma sonucu, hata kavramı, hata önleme yöntemleri, hata önleme yöntemlerinden biri olan kök neden analizleri, 7 kalite kontrol aracı, bu kontrol araçlarından Pareto ve sebep-sonuç grafiği hakkında bilgi verilmektedir.

Bölüm 3’te konu ile ilgili olarak literatürde yapılmış olan diğer benzer çalışmalara ayrıntılı bir şekilde değinilmektedir.

Bölüm 4’te konu ile ilgili yapılmış örnek bir vaka incelemesi bulunmaktadır.

Bölüm 5’te tez çalışmasının sonucu ve önerilere yer verilmektedir.

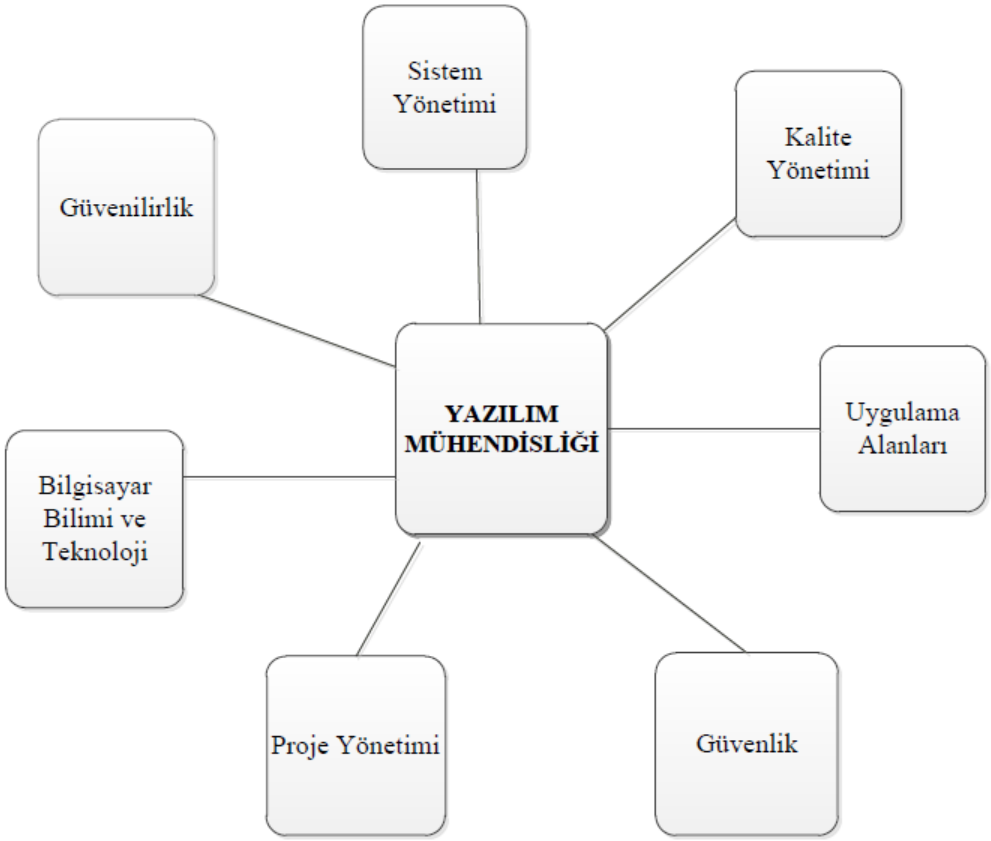
2. YAZILIM PROJELERİ VE HATA

2.1. Yazılım Projeleri

Günlük hayatta farkında olunmasa da, yazılımların insan hayatındaki yeri ve önemi gün geçtikçe artmaktadır. Yazılımlar ve yazılımların kullanıldığı araçlar insan hayatı ile o kadar bütünleşmiştir ki günlük hayat onlarsız yaşanmaz bir hal alabilmektedir. Yazılımlar, ev araçlarından uzay gemilerine kadar, birçok farklı alanda kullanılmaktadır. Örneklere bakılacak olunursa, bankacılık sektöründe işe uygun bir yazılım kullanmayan bir finans kurumunu düşünmek bile olası değildir. Sadece bankacılıkta değil telekomünikasyon, lojistik, perakende satış gibi hemen hemen her sektörde yazılım artık "olmazsa olmaz" bir bileşen haline gelmiştir. Sivil sektörlerde hal böyle iken savunma sektöründe de durum farklı değildir. Özellikle savunma sistemlerinde "yerli" yazılım üretilmesi son birkaç yıldır ülkemizin milli stratejisi olarak gündemde yerini korumaktadır. Hayatın bu kadar içine işlemiş olan yazılımların geliştirilmesi ve işler hale getirilmesi çok kolay olmamaktadır [1]. Bir yazılımın kalitesi ise, üretilen yazılımda az hata olması, kullanıcı/müşteri gereksinimlerinin tam olarak karşılanması, arızalar arasında geçen zamanın uzunluğu, arızaların giderilme hızı gibi ölçütler ile ölçülmektedir [2].

Çoğu kişi yazılımları bilgisayar programları olarak görmektedir. Aslında yazılım sadece bir bilgisayar programı değil, bu programın dokümantasyonunu ve doğru çalışması için gerekli olan bilgi başta olmak üzere daha birçok şeyi kapsamaktadır. Yazılımlar, tüm ürünün dokümantasyonu sayesinde tam olarak ortaya çıkmaktadır. Oluşturulan dokümanların bir kısmı ürünü açıklayan, bir kısmı müşterinin karşılaşacağı sorunlarda yararlanması için, bir kısmı ortaya çıkan yazılımın kurulumunu kolaylaştırması amaçlı hazırlanmıştır [3]. Bunun haricinde, bir yazılım sistemi, genellikle birçok ayrı programdan oluşmaktadır. Bu programlar yazılımın her bir fonksiyonunu içermektedir [4].

Yazılım projeleri diğer alanlardan bağımsız gibi görünse de gerçekte birçok alan ve dalla ilişkilidir. Şekil 1'de yazılım mühendisliğinin ilgili olduğu diğer alanlar verilmektedir [5]. Yazılım mühendisliği projeleri diğer alanlar sayesinde tamamlanır ve pekiştirilir.



Şekil 1. Yazılım Mühendisliği ve İlişkili Olduğu Diğer Alanlar

Yazılım projeleri diğer proje çeşitlerinden yapısal olarak oldukça farklıdır. Diğer proje çeşitleri ile kıyaslandığında yazılım projelerinde müşteri istekleri (gereksinimler) çok daha fazla değişkenlik göstermektedir. Bu değişkenlik sayesinde proje kapsamında kaç saat çalışılarak işin bitirilebileceği tam olarak belirlenemez. Projede gereksinimler değiştiği sürece iş en baştan yapılmaya devam edecektir. Bazen bir parçanın yazılımı birkaç saat alırken, fark edilemeyen bir kod yazım hatası yüzünden bu süre günleri bulabilir. Dolayısıyla, yazılım projeleri bilinmezlikler ve zorluklarla doludur [6].

Brooks [7] yazılım projelerini diğer alanlardaki projelerden ayıran özellikleri ve yazılım projelerinde yaşanan sorunları ve nedenleri 4 ana başlık altında toplamıştır. Bunlardan ilki *karmaşıklıktır*. Yazılım varlıkları, belki de büyüklük itibariyle diğer insan yapılarından daha karmaşıktır çünkü yazılımların hiçbir parçası birbirinin benzeri değildir. Karmaşıklıktan, teknik sorunların yanı sıra

yönetimsel sorunlar da ortaya çıkabilir. Karmaşıklık genel bakışı zor hale getirir ki bu durum da kavramsal tutarlılığa engel olur. Karmaşıklık, bütün sıkıntılı noktaların bulunup kontrol altına alınmasını zorlaştırır. Karmaşıklık, öğrenme ve anlama konusunda o kadar büyük bir yük oluşturur ki çalışanların işten ayrılmaları projeyi bir yıkım haline getirebilir. İkinci olarak *uyumluluk* özelliği tartışılmıştır. Projelerde karmaşıklık ne kadar artarsa, programlar arasında bulunan uyum o kadar zorlaşır. Çünkü kullanıcı için yaratılan arayüzün arkasında birçok farklı program birbiriyle uyum içinde sorunsuz çalışmak zorundadır. Üçüncü olarak *değişebilirlik* özelliği verilmiştir. Yazılım varlıkları sürekli olarak değişim baskısıyla karşı karşıyadırlar. Bu, her ne kadar diğer projelerde de (örneğin binalar, arabalar ve bilgisayarlar) öyle olsa da, imal edilen şeyler imalattan sonra çok seyrek değişirler; ya daha sonraki modellerle yer değiştirirler, ya da asli özellikleri, aynı temel tasarımla üretilen sonraki kopyalarının içine gömülür. Otomobillerin geri çağrılması son derece nadirdir; piyasada satılmakta olan bilgisayarların değişmesi de pek sık olmaz. Fakat yazılım tamamen sanaldır ve sınırsızca değiştirilebilir. Son olarak *görünmezlik* özelliği verilmiştir. Yazılım soyuttur, yani diğer ürünler gibi görünmez ve gözde canlandırılamazdır. Yazılımlar son ürün ortaya çıkana kadar her aşamada soyut ilerlemektedir ve bu ürün genelde bilgisayar ortamında yazılan yazılımdır [8].

Yazılım mühendisliği projeleri çeşitli evrelerden oluşmaktadır. Bu evreler her şirkette ve proje yapısında farklı biçimlerde tanımlanabilmektedir. Fakat genel olarak bakıldığında; bazı temel evreler çoğu yazılım projesinde ortaktır ve aynı şekilde tekrarlanmaktadır. Ürün tanımlama evresinde, ürünün pazardaki yeri belirlenir ve analiz edilir. Ürün geliştirme evresinde, müşteriden gelen istekler ve gereksinimler belirlenir. Belirlenen bu gereksinimler doğrultusunda ürünün özellikleri ortaya çıkarılmış olur. Tasarım evresinde, ürün özelliklerine uygun bir biçimde nasıl bir tasarım yapılması gerektiğine karar verilir. Uygulama evresinde ise kodlama, dokümantasyon ve yazılan kodu kurum içi test etmeye odaklanılmaktadır. Doğrulama evresi ise; müşteriye teslim edilen ürünün müşteri taraflı testlerini kapsamaktadır [9].

Yazılım Mühendisliği projelerinde bulunan karmaşık ve bilinmez yapıdan dolayı 1987 yılında Boehm tarafından yapılan bir araştırmada, yazılım mühendisliği projelerinde kritik olan ve dikkat edilmesi gereken noktalar 10 madde ile verilmiştir [10].

Bu maddeler aşağıdaki gibidir;

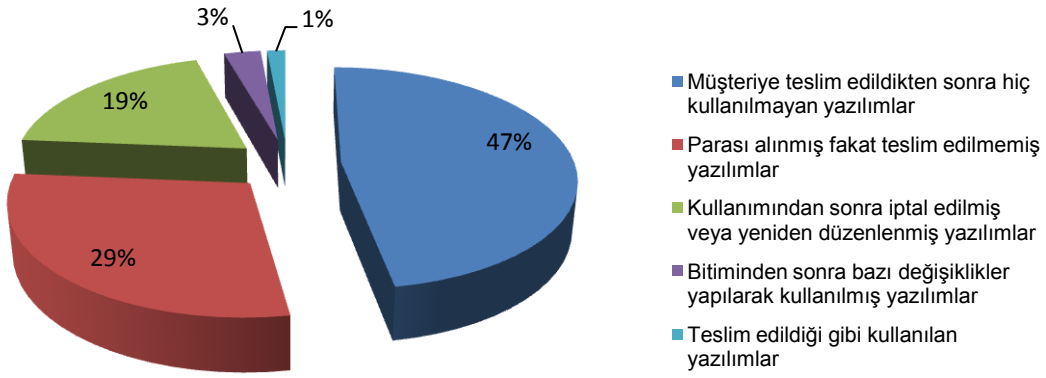
1. *Yazılımın müşteriye teslimatı yapıldıktan sonra bulunan hatanın maliyeti, gereksinim ve tasarım evresinde ortaya çıkan bir hatanın düzeltme maliyetinden 100 kat daha fazladır.* Dolayısıyla, projelerin gereksinim ve tasarım evrelerine yoğunlaşmak, hatta ek olarak bir prototip hazırlamak, hataların azalmasına ve projenin ilerleyen safhalarında çıkacak olan maliyetin düşmesine sebep olacaktır.
2. *Bir yazılım mühendisliği projesi maksimum projenin planlanan takviminin %25'i kadar erken tamamlanabilir*. Fazladan insan gücü eklemek dahi bu sayıyı azaltamayacaktır.
3. *Yazılım geliştirme evrelerine harcanan her 1 dolarlık bütçe karşılığında, bakım aşaması için 2 dolarlık bütçe* ayrılmalıdır. Proje yaşam döngüsü boyunca planlamanın bu durumu gözeterek yapılması, bakım aşamasında firmalara çok büyük kolaylıklar sağlayacaktır.
4. *Yazılım geliştirme ve bakım maliyetleri, öncelikli olarak ürünün içindeki kaynak talimatlarının sayısına bağlı bir değişkendir*. Bu durum alt düzey dillerden üst düzey dillere geçilmesindeki en önemli teşvik unsurudur.
5. *Projede çalışan kaynakların dolaşımı, yazılım üretkenliğini etkileyen en büyük faktörlerden biridir*. Araştırmalar, takım olarak yapılan çalışmanın, birey olarak yapılan çalışmaya kıyaslandığında üretkenlik değerinin 1/26' ya kadar çıktığını göstermektedir.
6. *Yazılım maliyetinin, donanım maliyetine oranı 1955 yılında 15/85 iken, bu oran (yazılım maliyeti /donanım maliyeti) 1985 yılında 85/15 olmuş ve gün geçtikçe artarak aynı yönde değişmektedir.* Bahsi geçen durum Brooks'un [7] makalesinde de belirtilmektedir.
7. *Toplam yazılım geliştirme eforunun %15'i programlama aşamasına ayrılmıştır. Yazılım geliştirilmeye başlanan ilk aşamalarda "40-20-40" (%40 analiz ve tasarım, %20 programlama ve kalan %40'lık dilim test ve entegrasyon) olarak ayrılan efor, günümüzde "60-15-25" olarak*

değişmiştir. Oranların bu şekilde değiştirilmesi, 1. maddede anlatılan maliyeti azaltma yöntemini de desteklemektedir.

8. *Yazılım sistemleri ve yazılım ürünleri, tek bir yazılıma göre 3 kat daha fazla maliyet oluşturmaktadır*. Dolayısıyla yazılım sistem ürünleri, tek bir yazılıma göre 9 kat daha fazla maliyetlidir.
9. *Proje esnasında yapılan ön kontroller hataların %60'lık bir bölümünü yakalamaktadır.* İyi bir yazılım denetimi veya yapılandırılmış bir ön kontrol maliyeti düşük olmasının yanında, yazılımda ortaya çıkan hataların azaltılmasında da rol oynamaktadır.
10. *Birçok yazılım mühendisi verileri takip etmek amaçlı Pareto grafiklerini kullanmaktadır. Bu grafikler %80'lik etkinin %20'lik kısımdan geleceğini söylemektedir.* Bunu bilmek projelerin %20'lik kısmına odaklanarak birçok hatanın önlenebileceğini göstermektedir. Örnek olarak;
    - Modüllerin %20'si, maliyetin %80'ini oluşturmaktadır.
    - Hataların %80'i, modüllerin %20'sinden kaynaklanmaktadır.
    - Bulunan hataların %20'si, hata düzeltme maliyetlerinin %80'ini oluşturmaktadır.
    - Modüllerin %20'si, uygulama zamanının %80'ini almaktadır.
    - Araçların %20'si, toplam araç kullanımının %80'ni teşkil etmektedir.

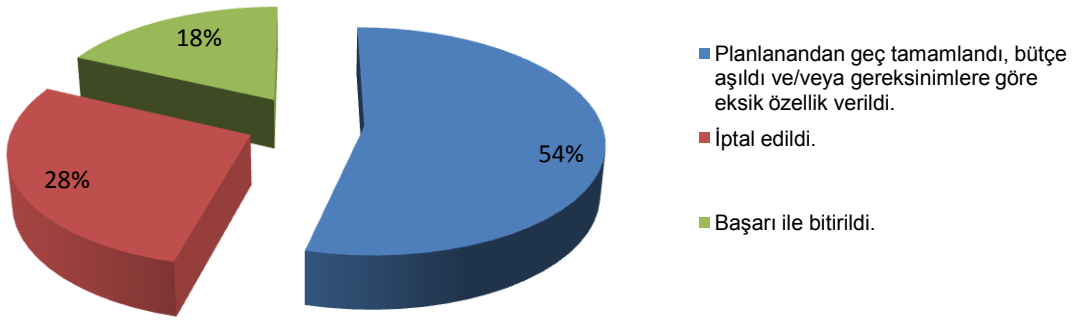
Pareto grafiklerine bölüm 2.6.1.'de ayrıntılı değinilecektir.

Brooks'un [7, 8] yazdığı "No Silver Bullet" isimli makalede yazılım projelerinin canavara dönüşebileceği söylenmektedir. Yazılım projeleri başlangıçta çoğunlukla "masum" görünüşlüdürler, başka bir deyişle her şey yolunda ilerliyor gibidir ama tutturulamamış planlara, aşılmış bütçelere ve hatalı ürünlere sahip bir canavara dönüşme özelliğine sahiptirler [7]. Yapılan yazılımların doğru ve eksiksiz çalışmalarının yanında bir diğer önemli konu ise müşterinin isteklerini ve gereksinimlerini karşılıyor olmasıdır. ABD'de yapılan bir araştırmaya göre doğru sistemi oluşturmanın, sistemi doğru oluşturmadan çok daha önemli olduğuna değinilmektedir. Bu araştırma ile ilgili elde edilen sonuçlar Şekil 2'de verilmektedir [11].



Şekil 2. Amerika Savunma Sanayii Sektöründe Yazılım Başarı Oranları

Şekil 2'den de görüldüğü üzere yazılım projelerinde başarı oranı düşüktür. Standish Group tarafından 1999 yılında yapılan araştırmaya göre 20 yazılım projesinin sonuçları incelenmiş ve projelerin başarı oranları hesaplanmıştır. İlgili araştırma raporuna göre ortaya çıkan sonuçlar Şekil 3'te verilmektedir [12, 13]. Şekil 3'ten görüldüğü gibi projelerde başarı oranı 3'te 1'den daha da azdır.



Şekil 3. Yazılım Projeleri Başarı Oranları

Başarılı olarak tanımlanan projeler, projenin zamanında, başlangıçta belirlenen bütçe ve fonksiyonellikte tamamlanan projelerdir.

Başarısız olarak tanımlanan projeler; projenin başında belirlenen özelliklerin bir kısmı tamamlanmayan ek olarak proje başında belirlenen bütçe ve teslim tarihinin aşıldığı projelerdir.

İptal edilen olarak tanımlanan projeler ise, projenin yaşam döngüsü esnasında herhangi bir sebepten dolayı tamamlanamamış projelerdir.

Şekil 3'te görüldüğü üzere, yazılım projelerinde başarı oranı %28 gibi düşük bir değer olarak bulunmuştur. Bu durumun en büyük sebebi, daha önceden de bahsedildiği gibi, yazılım projelerinin yapı olarak farklı ve karmaşık olmasıdır [14]. Bu başarının arttırılabilmesi için yapılabilecek birçok iyileştirme zaman içinde literatürde önerilmiştir. Bunlardan biri; proje yaşam döngüsü boyunca ortaya çıkan hataların tespit edilip ortadan kaldırılmasıdır [15].

İlerleyen yıllarda Standish Group tarafından bu çalışma tekrarlanmış ve değerler revize edilmiştir. 2006 yılında yapılmış olan araştırmaya göre başarıyla bitirilen proje yüzdesi %35'e çıkmıştır. Bu grup tarafından 1994, 1996, 1998, 2000 ve 2004 yıllarında yapılan araştırma sonuçları Tablo 1'de verilmektedir [16].

Tablo 1. Standish Group Tarafından Yapılmış Yıllara Göre Proje Değerlendirmeleri

Yıl	**Başarı %**	**Başarısız %**	**İptal Edilen %**
1994	16	53	31
1996	27	33	40
1998	26	46	28
2000	28	49	23
2004	29	53	18

Standish Group tarafından değerlerin düşüşü daha iyi proje yönetim süreci ve yöneticilerin projeye daha çok hakim olması olarak açıklanmıştır [13].

2.2. Hata Kavramı

Sektörü ne olursa olsun yapılan projelerin ortak amacı müşteriye eksiksiz ve hatasız bir şekilde istenilen ürünü sunabilmektir. Proje geliştirme süreci esnasında ne kadar çok kontrol yapılır bulunan hatalar düzeltilirse, müşteriye o kadar az hatalı ve kaliteli ürün teslim edilmiş olur.

Hata kavramı (defect) için literatürde birçok farklı tanım bulunmaktadır. Türk Dil Kurumu tarafından yapılan tanımlamada "istemeyerek ve bilmeyerek yapılan yanlış, kusur, yanılma veya yanılgı" [17] olarak verilen hata tanımı, Oxford sözlüğü tarafından "bir şeyin yarar veya değerini kaybetmesine neden olan şey" olarak yapılmaktadır [18]. Proje hangi sektörde yapılırsa yapılsın ortaya çıkan hata, sistemin yanlış ya da eksik çalışmasına neden olmaktadır.

Üretim sektöründe çeşitli ölçüm sonuçlarına göre istenmeyen değişikliklerin oluşması hata olarak kabul edilmektedir. Başka bir deyişle, önceden belirlenmiş olan spesifikasyon limitleri dışına çıkan uyumsuz veya kusurlu ürüne[1] hata adı verilmektedir [19].

2.3. Yazılım Projelerinde Hata

Hata kavramı kişinin perspektifine göre farklılık göstermektedir. Yazılımı kullanan kişiler gözünden bakıldığında, yazılımın son kullanıcıların beklentileriyle buluşmadığı her nokta birer hatadır [9]. Bu bağlamda son kullanıcı, yazılımı kullanan bir insan ya da başka bir yazılım olabilir. Yazılım mühendisi gözünden hata ise, üzerinde çalışılan yazılımda her değiştirilmesi gereken noktadır [11]. Yazılım projelerinde bulunan hatalar için, Tablo 2'de birkaç örnek verilmektedir [9].

Tabloda yer verilen hatalar küçük boyutlu ve düzeltilmesi kolay olan hatalara bir kaç örnek[2] barındırmaktadır. Bu tür hataların yanında etkileri çok daha büyük olan hatalar da bulunmaktadır. Yazılımın bu kadar hayatın içine işlemiş olması günlük hayatta sayılamayacak kadar fayda sağlasa da, örnekte olduğu gibi geri dönülemeyecek sonuçlar da doğurabilir.

---

[1] Örnek olarak; yarı iletken üretimi yapan bir fabrikanın ürettiği cisim için, üretim esnasında kitleden 25 örnek alınmıştır. Eğer bu 25 ürünün 5 tanesi belirlenen spesifikasyon limitleri içinde ise süreç istatistiksel olarak kontrol altında olacaktır. Spesifikasyon limitleri, örneklemden alınan değerler ile 1.69 ve 1.13 mikron olarak belirlenmiştir. Üretilen ürünün mikron ölçümleri bu değerler dışına çıkarsa ürün hatalı olarak kayıt altına alınacak, hata sebebi belirlenmek için süreç irdelenecektir. Yani üretim sektöründe önceden belirlenen özelliklere uymayan ürün hatalı üründür [41].

[2] Radyoterapi aracı 2 farklı modda çalışmak üzere tasarlanmıştır. İlk mod, az dozda radyasyon verirken 2. mod, daha küçük bir alana daha fazla radyasyon vermelidir. Hastaya ilgili dozaj uygulanırken maske takılması gerekmektedir. Raporlanan veri; hastaya yüksek dozda radyasyon verilirken vücudun diğer bölgelerine koruyucu maske kullanılmamasından dolayı hastanın öldüğü yönündedir. Resmi sonuçlara göre, radyoterapi makinasını kontrol eden bir insan değil, bir yazılım sistemidir. Bu yazılımın özellikleri arasında, ilgili yere radyasyon verilirken, radyasyonun diğer bölgeleri etkilememesi adına bir kontrol mekanizması bulunmaktadır. Fakat yazılımın kendini kilitlemesinden dolayı radyasyon her bölgeye verilmiştir. Hastanın ölüm sebebi makinanın yazılımında gerçekleşen bir hatadan kaynaklanmıştır [11].

Tablo 2. Kullanıcı Perspektifinden Yazılım Hatalarına Birkaç Örnek

Tipik yazılım hataları	
Kullanıcı beklentisi	Yazılım, işi yapmamda yardımcı olacak.
Yazılım Hatası	İstenilen yazılım fonksiyonelliği yok.
Kullanıcı beklentisi	Düğmeye basmak yapmak istediğim işi gerçekleştirecek.
Yazılım Hatası	Düğmeye basmak hiç bir şey yapmayacak ya da istediğim şeyi yapmayacak.
Kullanıcı beklentisi	Bir dosya başka bir yere başarıyla kopyalanabilir.
Yazılım Hatası	Dosya kopyalama aşamasında bozuldu.
Daha az belirgin yazılım hataları	
Kullanıcı beklentisi	Yazılım hatalardan kaçınmama yardımcı olacak (örneğin, yazım hataları).
Yazılım Hatası	Geçerli bir sözcüğü hatalı olarak kullanmaktan kaynaklanan bir yazım hatası algılanmadı.
Kullanıcı beklentisi	Yazılım hızlı bir şekilde cevap verecektir.
Yazılım Hatası	Kullanıcı bakış açısından yazılım yavaş cevap veriyor.
Kullanıcı beklentisi	Yazılım güvenli.
Yazılım Hatası	Kişiller güvenlik açıklarından yararlanıp yazılıma saldırabilir.

Yazılım mühendisliği projelerinde kayıt altına alınan hatalar, hatanın oluştuğu duruma göre gruplandırılabilir. Bu gruplandırmalar sayesinde hatanın kaynak noktası daha rahat tespit edilerek, süreçteki eksiklikler giderilebilir böylece süreçte tekrar aynı hataların yapılması önlenebilir. Sonuç olarak süreç iyileştirme yapılmış olur [20]. Yazılım mühendisliği projelerinde hataların gruplandırıldığı kategoriler yazılıma, araştırmaya veya şirkete göre değişmektedir. Örnek olarak; sistem verisinin istenilen özelliklerde çalışmamasından kaynaklanan hatalar, test parametrelerinde unutulan ya da yanlış test edilen alanlardan kaynaklı hatalar, dokümantasyon sırasında ortaya çıkan hatalar, parçaların birbirleri arasında iletişim sorunu yüzünden ortaya çıkan hatalar, yazılan kodun performansından kaynaklanan hatalar, yazılan algoritmalardan kaynaklanan hatalar şeklinde kategorilere ayrılabilmektedir [2].

2.4. Hata Önleme Yöntemleri

Hata önleme yöntemlerinin temel amacı ortaya çıkarılan ürünün kalitesinin ve verimliliğinin arttırılmasıdır. Hata analizleri, projelerin yaşam döngüleri boyunca toplanan verilerden elde edilir. Toplanan hata verileri, süreci ve proje çalışanlarını geliştirme amaçlı kullanılmaktadır. Hata analizleri sayesinde bulunan hataların kök sebepleri belirlenir ve projelerin geliştirilme evrelerinde hataların azaltılması için çalışmalar yapılır. Hata analizlerinde sınıflandırma yöntemleri kullanılmaktadır. Bu analizlerden ikisi kök neden analizleri ve stokastik modelleme yöntemleridir. Stokastik modelleme yöntemi; olasılıksal eşitliklerle bir olayın daha sonraki evrelerinde nasıl davranış göstereceği konusunda tahminleme yapmaktadır [20].

2.5. Kök Neden Analizleri

Kök neden analizleri; problemlerin kalite, güvenilirlik ve çevresel etkilerini belirlemek için kullanılan bir analiz çeşididir. Başka bir deyişle, performans problemlerinin sebeplerini bulmak ve düzeltmek için kullanılırlar [21]. Bu analizde amaç; bulunan hataların çeşitlerini ve sebeplerini ortaya çıkarmak ve çözüm bulmaya çalışmaktır [22]. Analiz sonucunda elde edilen diyagram, kalite karakteristikleriyle etmenler arasındaki ilişkiyi gösteren bir diyagramdır [23]. Kök neden analizi, süreçlerin içindeki çabaları ve sebeplerini mümkün olduğu kadarıyla açıkça tanımlamayı sağlamak için yakın detay örnekleri olabilenlerinin elde edilmesini gerektirebilir [22].

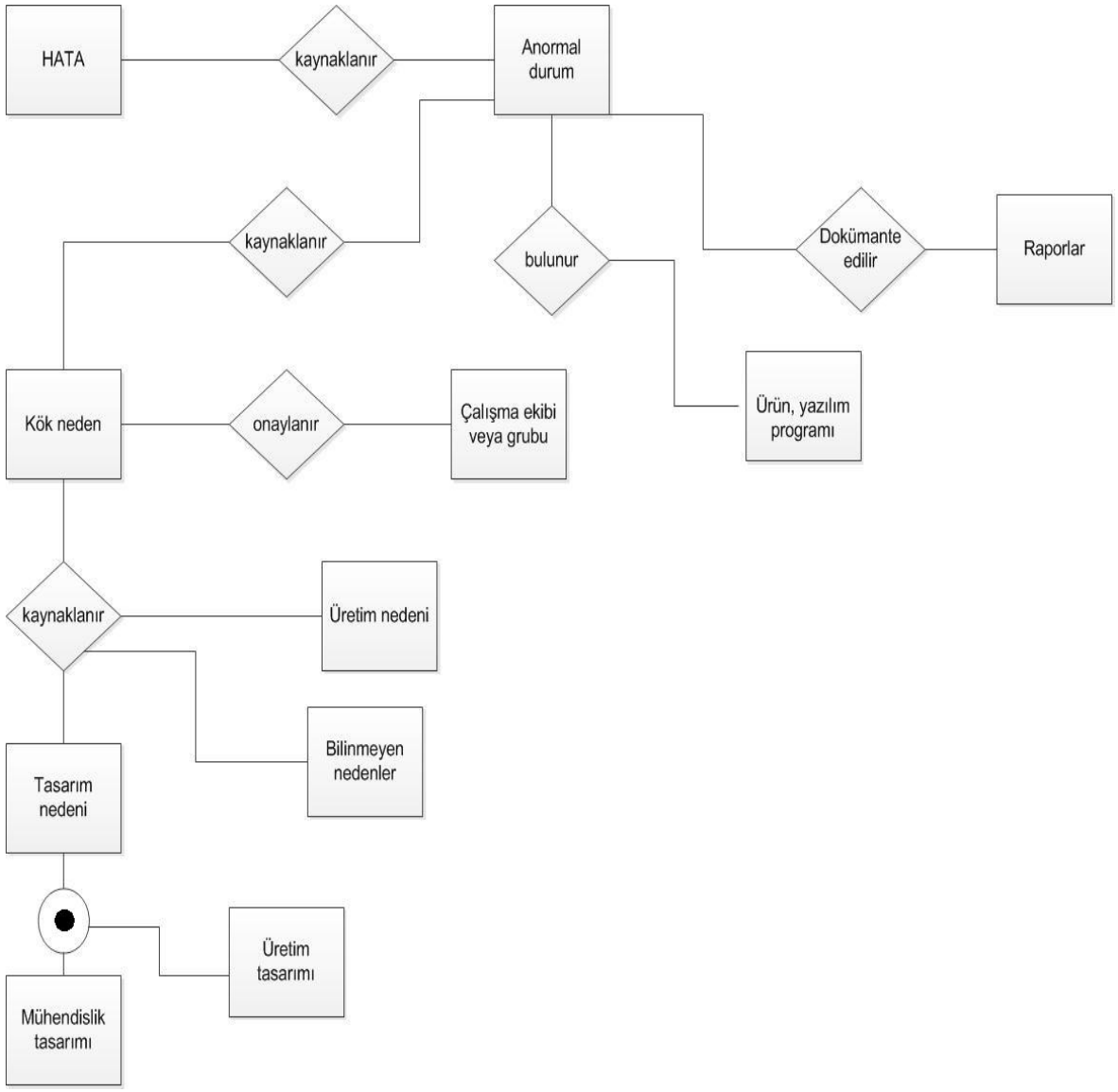
Kök nedenler, bir problemin arkasında yatan gerçek sebeplerdir. Bu analiz, balık kılçığı grafiği olarak adlandırılır ve yaygın olarak kullanılır [22]. Kök neden analizleri genellikle literatürde Pareto grafikleri ile desteklenmektedir. Bu grafikler problemlerin ve problem sebeplerinin özetlendiği grafiklerdir [15]. İlgili grafik çeşitleri ilerleyen bölümlerde daha ayrıntılı anlatılacaktır.

Erhan Uysal'ın yaptığı bir araştırmada [24], projelerde herhangi bir problem veya olay için kök neden analizi yapılma nedenini şu şekilde sıralamıştır;

- Problemlerin gerçek çözümlerine ulaşmak, problemin varoluş nedenlerini öğrenmeyi sağlamak,

- Probleme nelerin sebep olduğunu analiz etmeden iyileştirmeler planlamak ve gerçekleştirmek, elemanların ve zamanın yanlış ve sorumsuzca kullanılması ile sonuçlanmaktadır. Bu oluşacak zaman kaybını önlemek,
- Sorunların doğru anlaşılmasını sağlayarak kurumda sahiplenmeyi arttırmak,
- Kök neden analizleri sonucu, bilgi seviyesinin, farkındalıkların ve davranış biçimlerinin değişmesi ile hedeflerin de olumlu yönde değişmesinin sağlamak, olarak verilmektedir [24].

Kök neden analiz çalışması ve kök sebeplerin belirlenmesi Şekil 4'te kısaca özetlenmiştir [25].



Şekil 4. Kök Neden Analizi

2.6. Kalite Kontrol Araçları

İstatistiksel analiz işlemlerinin başlangıç noktası verilerin toplanması, tanımlanması ve özetlenmesidir. Elde edilen verilerin anlamlı hale getirilmesi ve yorumlanabilmesi için çeşitli araçlar ve grafikler kullanılmaktadır [26]. Kalite kontrol elemanlarının süreçte karşılaşabileceği problemlerin çözümleri için aşağıda belirtilen meşhur yedi istatistik teknik kullanılır [27]. Bu yedi kalite aracı kök neden analizlerinde de kullanılacak olan verinin belirlenmesinde büyük rol oynamaktadır.

Bu yedi kalite aracı aşağıdaki gibidir;

1) Pareto Diyagramı
2) Kontrol Çizelgesi
3) Kontrol Diyagramı
4) Histogram
5) Sebep-sonuç Diyagramı (Balık kılçığı diyagramı)
6) Serpilme Diyagramı
7) Akış Diyagramı

Yukarıda verilmekte olan yedi kalite kontrol aracı sayesinde, toplam kalite konusunda Japonya'daki en önemli isimlerden biri olan Kaoru Ishikawa, bir işletmedeki problemlerin %95'inin bu araçlar yardımıyla çözülebileceğini savunmaktadır [28].

Yapılacak olan analiz çeşidine göre, yedi kalite aracından uygun olan grafik tipi seçilir. Bu seçim, analizde bulunacak olan veri tipine ve elde edilmek istenen grafiğe dayanarak yapılmalıdır. Bölüm 4'te yapılacak olan vaka çalışması kapsamında yukarıda listelenmekte olan yedi araçtan Pareto ve sebep-sonuç diyagramları kullanılacaktır. Bunun sebebi; kök neden analizi kapsamında Pareto ve sebep-sonuç grafiklerinin destekleyici olmasıdır.

2.6.1. Pareto diyagramları

Projeler boyunca toplanan binlerce verinin düzenlenmesi ve bu veriler hakkında yorum yapılması oldukça zordur. Pareto analizi, verileri tasnif ederek karar alma işini kolaylaştırır.

Vilfredo Pareto tarafından ortaya çıkarılmış olan ve 80 / 20 kuralı olarak bilinen Pareto prensibi, Pareto analizi yönteminin temelini oluşturmaktadır [27]. Normal dağılımda sebeplerin en önemli %20'si, sonuçların %80'ini, sonra gelen %30'u, sonuçların %15'ini ve geri kalan %50'si ise sonuçların sadece %5'inin oluşmasına neden olmaktadır. Maliyetin yaklaşık %80'ninin kişilerin sadece %20'sinden kaynaklandığı veya servetin yaklaşık %80'ninin nüfusun %20'sinin elinde olduğu gibi durumlar da bu konuya birer örnektir[3]. Başka bir deyişle, Pareto prensibi genellikle problemlerin ya da kusurların büyük bölümünün oldukça az sayıda nedenden kaynaklandığını söylemektedir [29].

Oluş sıklığına göre düzenlenmiş özel bir histogram türü olan Pareto grafiği, değişik sayıdaki önemli sebepleri, daha az önemde olan sebeplerden ayırmak için kullanılan bir tekniktir. Bu teknik bir olayın grafik yardımıyla gösterilmesi ve karşılaşılan problemin veya konunun en önemli sebebi üzerinde dikkati yoğunlaştırdığından ve önceliklerin belirlenmesine yardımcı olduğundan her alanda bulunan verinin incelenmesine olanak sağlamaktadır [27, 29]. Pareto prensibi göstermiştir ki, yaşanan problemlerin büyük çoğunluğu az sayıdaki önemli faktörden kaynaklanmaktadır. Eğer bu ana sebeplerin yol açtığı sorunlar düzeltilebilirse başarı şansı artacaktır. Pareto analizi, yöntemi uygulayacak takımların sorunlara yol açan ana sebepler üzerinde hızlı bir biçimde odaklanıp problemi tespit edebilmeleri için kullanılır [30].

Pareto grafikleri ile tüm veri içinden, üstünde kök neden analizleri yapılması planlanan uç veriler tespit edilir. Bulunan bu uç veriler sebep-sonuç grafikleri yardımıyla daha da irdelenir. Bölüm 4'te yapılacak olan vaka incelemesi kapsamında hataların sebepleri Pareto grafikleri yardımıyla bulunacaktır.

2.6.2. Sebep-sonuç diyagramları

Sebep-sonuç diyagramı belirli bir problemin veya durumun olası nedenlerini belirlemek için kullanılmaktadır. İstatistiksel yöntemler kullanarak sonuçlardan hareketle sebeplere ulaşabildiğimize göre, sonuçlarla bunları doğuran sebepler arasındaki çapraz ilişkinin ortaya çıkarılması ve görsel olarak sunulması

[3] Vilfredo Pareto ilk olarak İtalya topraklarının % 80'ine nüfusun % 20'sinin sahip olduğunu fark ederek bu prensibi ortaya atmıştır [27].

gerekmektedir. Bunu en kolay sebep-sonuç diyagramları ile yapabiliriz [31]. Çizilen grafiğin görüntüsü balık kılçığını andırdığından, bu grafik balık kılçığı grafiği olarak da adlandırılmaktadır.

Bu grafik çeşidi sayesinde;

- Problem çözme sürecinin daha düzenli hale getirilmesi,
- Problem hakkında bütün bilinenlerin ortaya konması, bilinenlerden bilinmeyene doğru sistematik bir yaklaşım olması,
- Problemle doğrudan deneyimi olan kişilerin uzmanlığından yararlanılması,

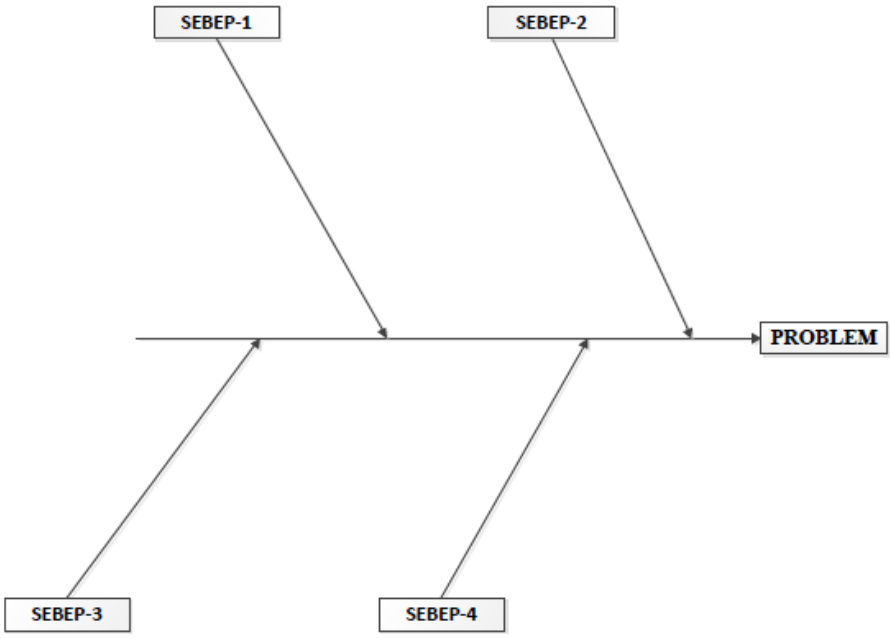
mümkün olabilmektedir [28].

Özden'in [32] tanımlamış olduğu ve sebep–sonuç diyagramlarını oluştururken izlenilecek olan aşamalar aşağıda yer almaktadır;

1. Dağılım analizinde, önce geliştirilmesi amaçlanan problem belirlenmelidir [31]. Problem belirlendikten sonra Şekil 5'te verildiği üzere sebep-sonuç diyagramının ana çizgisi oluşturulur ve çözümü bulunması istenen ana problem yazılır.

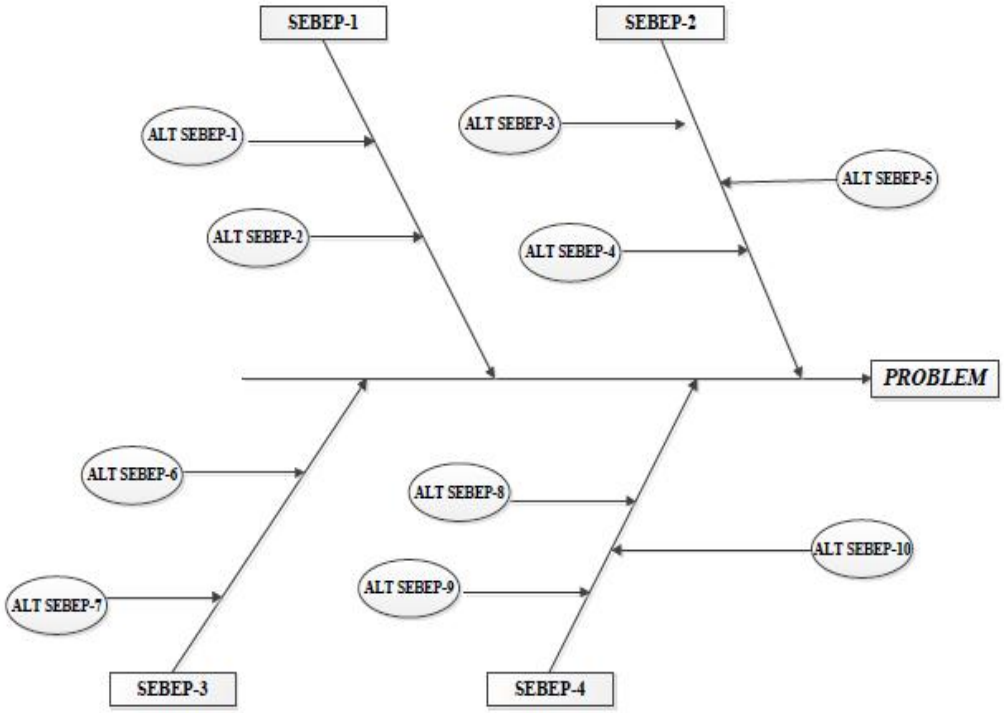
Şekil 5. Sebep-Sonuç Aşama-1

2. Diyagram oluşturulurken, çevresel ve işletme içi faktörler ayrıntılı olarak incelenmelidir [31]. Problemin kaynaklandığı muhtemel sebepler belirlenerek ana başlıklar altında toplanmalıdır. Tüm sebeplerin aktarılabilmesi için problemle ilgili olan herkesin görüşü alınmalıdır. Üretim sektöründe genellikle belirlenen bu ana sebepler Metot, Makine, İnsan gücü ve Malzeme olarak gruplandırılmaktadır. Fakat yapılan analizlere göre bu sebepler farklılaşabilmektedir. Belirlenen ana sebepler Şekil 6'da gösterildiği gibi grafiğe eklenir.



Şekil 6. Sebep-Sonuç Aşama-2

3. Proje kapsamında olan tüm üyeler beyin fırtınası yöntemiyle bulunan bu ana sebeplerin alt sebeplerini belirler. Bu aşamada önemli olan nokta, beyin fırtınasına katılan kişilerin proje ve problem hakkında bilgi sahibi olmalarıdır. Bu sayede gerçek sebeplere ulaşılabilmektedir. Şekil 7'de grafiğin bu evreden sonraki durumu verilmektedir.



Şekil 7. Sebep-Sonuç Aşama-3

4. Sebep-sonuç diyagramlarının çizilmesinde yapılan son aşama; belirlenen ana sebeplere uygun olarak yazılmış olan alt sebepler beyin fırtınasına katılan kişiler tarafından değerlendirilir. Kişiler teker teker, kendi önem sıralarına uygun puan verirler. Puanlama sonucunda, her alt sebebe kişilerin belirledikleri toplam not verilir. En çok oyu alan alt sebep belirlenir, kök neden analizine göre problemin kaynak noktası en çok oyu alan alt sebeptir [32]. En çok oyu alan bu alt sebepten sonra iyileştirme ve sebebi ortadan kaldırmaya yönelik çalışmalar yapılmalıdır. Zaman içinde elde edilmiş diyagramın güncelleştirilmesi gerekmektedir [31].

3. LİTERATÜR TARAMASI

Şirketlerde ortak amaç müşteriye teslim edilecek olan son ürünün en az, hatta sıfır hata ile teslim edilmesidir. Hataları en aza indirebilmek için, sektörden bağımsız olarak yapılmış birçok farklı araştırma mevcuttur. Bu çalışma kapsamında, farklı sektörlerde yapılmış olan benzer araştırmalar incelenmiştir. Bölüm 2.2'de değinildiği gibi, hata kavramı, çalışılan sektörün özelliklerine göre değişiklik göstermektedir. Dolayısıyla sektörlere göre süreç içerisinde bulunan hata kavramları birbirinden farklılaşsa bile, kullanılan yöntemler benzerlik göstermektedir.

3.1. Bircan ve Gedik'in Çalışması

Bircan ve Gedik [33] tarafından yapılan çalışmada, Sivas Dikimevi'nde üretim yapan bir şirketin 1 Ocak 2001 ve 30 Haziran 2001 tarihleri arasında tespit edilen hatalarının sebepleri araştırılmıştır. Üretim esnasında toplanan veriler incelenmiş ve meydana gelen hataların önceden belirlenen spesifikasyonlara uygun olup olmadıkları araştırılmıştır. Veriler 24 haftada 75 adet, toplam 1800 adet rüzgar ceketi[4] hataları ile ilgilidir. Veriler kullanılarak Pareto ve sebep-sonuç diyagramları ve hata yoğunluk diyagramı ile ilgili uygulamalar yapılmıştır. Pareto diyagramı sayesinde problemlerin hangisinin öncelikle ele alınması gerektiği tespit edilmiştir. Araştırma sonucuna göre ortaya çıkan Pareto grafiğinde sayı ile belirtilen hata kategorilerinin isimlerinin açık bir şekilde gösterimi ve dağılım sayıları Tablo 3'te verilmektedir [33].

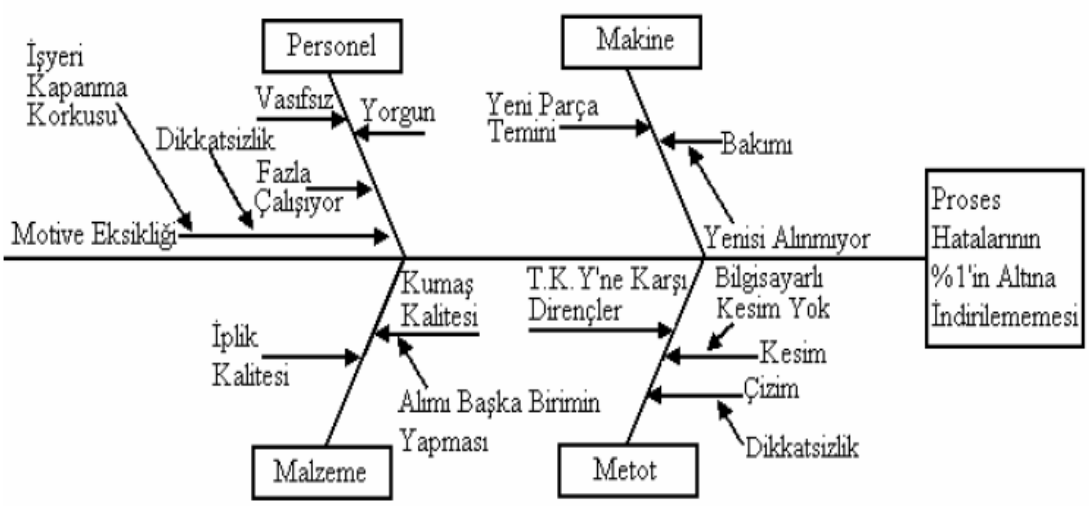
[4] Kişiyi açık havada, soğuktan ve dış etkilerden koruyan giyecek.

Tablo 3. Sivas Dikimevi'nde Bulunan Hata Sıralama Dağılımı

NO:	**Hata Türü**	**Hatalı Adet**	**%**
1	Fermatüre montesi ve dikişler	5	18
2	Göğüs cep ve kapak takma	4	15
4	Ölçüler	3	11
5	Makine ayarı ve dikiş aralıklar	3	11
6	Yaka takma ve üst baskısı	3	11
7	Kumaş ve Renk uyumu	2	7
8	Kol takma ve apara kaçığı	2	7
9	Baskı dikişleri	2	7
10	Bel kordonu montesi	2	7
11	Etek katlama ve kapsül basma	1	3
12	Fermuar montesi	1	3
TOPLAM		**28**	**100**

Tablo 3'te görüldüğü üzere ilk 2 hata, hataların 1/3'üne karşılık gelmektedir. Araştırmada öncelikle hata oranlarının büyükten küçüğe doğru çözümüne ulaşılmaya çalışılmıştır.

Yapılan beyin fırtınası sonucu "malzeme", "personel", "metot" ve "makine" maddeleri ana sorunlar olarak seçilmiştir. Daha sonra bu verilere göre sebep-sonuç diyagramı hazırlanmıştır. Hazırlanan sebep-sonuç diyagramından kök sebepler tespit edilmiştir. Bu sayede sorunun muhtemel nedenleri belirlenmiştir. Çalışma kapsamında oluşturulan sebep-sonuç diyagramı Şekil 8'de verilmiştir [33].



Şekil 8. Sivas Dikimevi'nde Bulunan Hataların Azaltılması Durumu için Sebep-Sonuç Diyagramı

Sivas Dikimevi'nde yapılan araştırmanın yazılım sektöründe yapılan araştırmalardan bir farkı; üretim sektöründe sıkça kullanılan hata yoğunluk diyagramlarının kullanılmasıdır. Yazılım projelerinde bulunan hatalar için Pareto grafikleri yardımıyla ilgili grafik sonucu üretilebilmektedir.

Araştırma kapsamında oluşturulan hata yoğunluk diyagramı[5], problemi meydana getiren nedenlere inmek için etkin bir araçtır. Verilerin tamamına bakıldığında sorun açıkmış gibi görünür, fakat veriler daha küçük parçalara ayrılmadıkça güçlüğün ne olduğunu belirlemek oldukça zordur. Hata yoğunluk diyagramıyla bir bütünü parçalara ayırarak her bir parçayı daha iyi inceleyebiliriz. Hata yoğunluk diyagramı tek başına bir sorunu çözmeyebilir, ama çözüme ulaşmada yardımcı olur. Kesim sonrası rüzgar ceketlerinin kontrolünde belirlenen hataların, hata yoğunluk diyagramı bu şekil üstünde oluşturulmuştur. 10.06.2001-20.06.2001 tarihleri arasında kesim sonrası incelenen 200 adet rüzgar ceketinde 6 adet hata tespit edilmiştir. Sonuç olarak hataların özellikle yaka kısmında olup, kesim, ölçü,

[5] Hata yoğunluk diyagramı, mamul maddeyi çeşitli açılardan gösteren bir resimdir. Mamulün görünen kısımlarına ait resimlerini ihtiva eden bu diyagramın üzerine hataların tipleri işaretlenir. Her bir ürünün tek tek muayenesi sonucunda kusurların nerelerde yoğunlaştığı gözlenir ve bu kusurlar diyagramda ilgili yerlere işaretlenir. Gereksiz hata çeşitleri kategorilere ayrılarak her bir hata farklı renkte, sembolde veya desende gösterilebilir. Böylece mamulün neresinde veya hangi bölgesinde ne tip kusurların yoğunlaştığı belirlenerek üretim sürecinde bunların önlenmesine dönük tedbir alınır [44].

ilik hatası ve renk uyumsuzluğundan kaynaklandığı tespit edilmiş, personel uyarılarak bu hataların önlenmesine gidilmiştir [33].

3.2. Kumaresh ve Baskaran'ın Çalışması

Yazılım alanında bugüne kadar yapılmış olan çalışmalar genellikle hata önleme yöntemleri üzerine odaklanmıştır. Kumaresh ve Baskaran [20] tarafından 2010 yılında yapılan bir çalışmada hatalar üzerinde çeşitli değerler toplanmış ve araştırmalar yapılmıştır. Araştırma kapsamında belirlenen 5 farklı projede toplam 637 hata tespit edilmiştir. Tespit edilen hatalar, hataların kaynaklandığı faza göre gruplandırıldığında en fazla hatanın 420 hata ile kodlama aşamasında yapıldığı görülmüştür. Hesaplanan değerler Tablo 4'te verilmektedir [20].

Tablo 4. Kumaresh ve Baskaran Araştımasında Bulunan Hata Değerleri

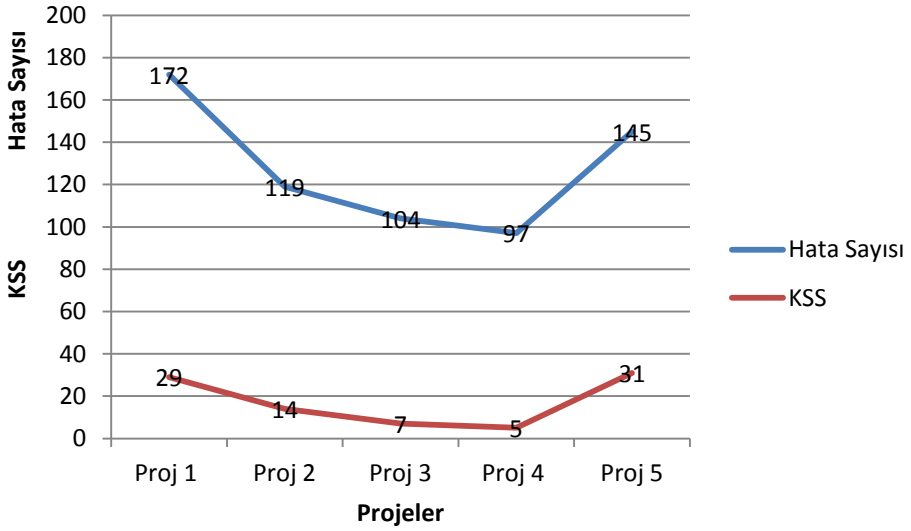
Proje No	**Proje Adı &Tanım**	**KLOC**	**Hata Sayısı**	**Efor (saat)**	**Hata Yoğunluğu**
Proje 1	Maple - CRM module for a trading company	29	172	2200	0.006
Proje 2	Indesign – Survey Automation Tool	14	119	1200	0.009
Proje 3	Stock Market Application	7	104	600	0.015
Proje 4	Issue Tracker Manages and maintains list of issues raised by an organization	5	97	400	0.019
Proje 5	GRTNET - General Reporting Tool	31	145	2400	0.005

Tablo 4'te bulunan Hata Yoğunluğu (Defect Density) değeri projede tespit edilen toplam hata sayısının, yazılım geliştirilirken ortaya çıkan bin kod satır sayısına[6] bölümünden oluşmaktadır.

$$\text{Hata Yoğunluğu} = \frac{\text{Hata Sayısı}}{\text{Büyüklük (kloc)}} - 1 \qquad (3.1)$$

[6] Kod Satır Sayısı (SLOC) yazılımın kaynak kodunda bulunan kod satır sayısının sayılması ile oluşan bir yazılım metriğidir.

Projelerin büyüklüklerine göre hata yoğunluk değerleri ise Şekil 9'da bulunmaktadır. Şekil 9'da görüldüğü üzere hata sayısı ve proje büyüklüğü arasında doğru orantı bulunmaktadır. Proje büyüdükçe hata sayısı da buna orantılı bir biçimde artmaktadır.



Şekil 9. Kumaresh ve Baskaran Araştımasında Bulunan Hata Sayısı-Proje Büyüklüğü İlişki Grafiği

Araştırmanın bir sonraki aşamasında DHS yöntemi[7] kullanılarak hataların tipleri ve sınıfları tespit edilmiş, belirlenen bu sınıflara göre hatalar kategorize edilmiştir. Kategorize edilen hata tiplerini kodları ve açıklamaları Tablo 5'te verilmektedir.

[7] DHS, yazılım geliştirme süreci içerisinde hem hataları bulma etkinliklerinin düzenlenmesindeki sorunları, hem de bulunan hataların sürecimizin gelişmesi yönünde fayda sağlaması için kullanılan metotların dezavantajlarını ortadan kaldıran çözümler sunar. DHS, hatalardan anlamsal bilginin çıkarılması ile geliştiriciye süreç içinde erken aşamalarda hızlı bir şekilde geri bildirimlerde bulunmayı amaçlar [35].

Tablo 5. Kumaresh ve Baskaran Araştımasında Bulunan Hataların Kodları ve Alanları

Kod	**Adı**	**Açıklaması**
MAN	Mantık Hatası	Mantık Hatası
GER	Gereksinimler	Gereksinimlerin yanlış anlaşılması ya da tanımlanırken eksik tanımlanması
GH	Grafiksel Hata	Ekran, rapor ve ya tasarıma bulunan hata
TSRM	Tasarım Hatası	Tasarımın yetersiz yapılmasından kaynaklanan hata
DOK	Dokümantasyon	Doküman yazım hataları, kod yazım hataları

Belirlenen hata tipleri, bu hata tiplerine karşılık gelen hata sayıları ve hataların bulunduğu proje fazları Tablo 6'da verilmektedir.

Tablo 6. Kumaresh ve Baskaran Araştımasında Bulunan Projeler için Hata Kategorizasyonları

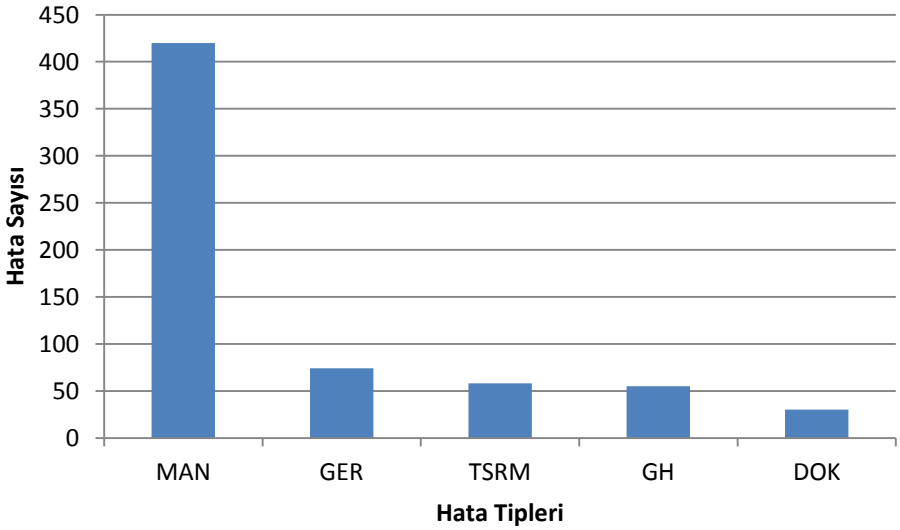
Faz Yaşam Döngüsü Fazı	**Aktivite**	**Hata Tipi**	**Hata Sayısı**
Gereksinimler	Gözden geçirme	GER	74
Tasarım	Gözden geçirme	TSRM	58
Kod	Test (Fonksiyon/Birim)	MAN	420
Grafiksel Hata	Gözden geçirme	GH	55
Dokümantasyon	Gözden geçirme	DOK	30

Tablo 6'da DHS yöntemiyle belirlenen hataların, analiz kapsamında bulunan 5 farklı projeye dağılımı Tablo 7'de verilmektedir.

Tablo 7. Kumaresh ve Baskaran Araştımasında Kullanılan Projelere ve Hata Tiplerine Göre Hata Sayıları

Proj No	GER	TSRM	MAN	GH	DOK	TOTAL
Proje 1	20	17	120	9	6	172
Proje 2	15	11	75	10	8	119
Proje 3	12	14	58	13	7	104
Proje 4	12	8	60	15	2	97
Proje 5	15	8	107	8	7	145
TOTAL	**74**	**58**	**420**	**55**	**30**	**637**

Bir sonraki aşamada Tablo 7'de verilen değerler kullanılarak Pareto grafiği çizilmiştir. Çizilen Pareto grafiği Şekil 10'da verilmektedir.



Şekil 10. Kumaresh ve Baskaran Araştımasında Oluşturulan Pareto Grafiği

Şekil 10'da veriler Pareto grafiğine göre analiz kapsamında bulunan 5 proje için en çok hatanın Mantık (MAN), Gereksinimler (GER), Tasarım (TSRM) kodları ile verilen hata tiplerinden kaynaklandığı görülmektedir. Bu 3 hata tipinde bulunan hata sayısı toplam hata sayısının %80'ini oluşturmaktadır. Dolayısıyla kök neden analizi kapsamında önem verilmesi gereken ve öncelik derecesi yüksek olan veriler bu 3 kategoriden kaynaklanmaktadır. Çalışmanın devamında, yapılan kök neden analizi ile bir sebep-sonuç grafiği elde edilmiştir.

Sebep-sonuç grafiği yardımıyla ortaya çıkan hata sebeplerini önlemek için çeşitli planlar belirlenmiştir. Hata tipi MAN olarak tanımlanmış hataların önlem planı olarak, proje çalışanlarına eğitim verilmesi, test ve kodlama aşamalarında eğitilmiş veya tecrübeli personelin kullanılması olarak belirlenmiştir. GER olarak tanımlanmış olan gereksinim tipli hatalarda, müşteriyle gereksinimlerin belirlenmesinden sonra bir daha değişmeyeceğine dair imza alınması kararı alınmıştır. TSRM hatalarında, doğru aracın seçilmesi ve eğitiminin verilmesi konusunda çalışma yapılması kararı alınmıştır. GH hata tipi için aynı tip projelerin grafik kontrollerinin yapılması için bir araç geliştirilmesine ve bu aracın proje başlamadan önce işlemi otomatik olarak yapmasına karar verilmiştir. DOK hata tipi ile bulunan veriler için ise, müşteriye ürün teslimi yapılmadan önce tüm verinin kontrolünün yapılmasına karar verilmiştir.

Bir sonraki aşamada belirlenen önleyici yöntemler, projelerde uygulanmış ve bulunan hata sayılarındaki değişim tespit edilmiştir. Tekrar ölçülen değerler Tablo 8'de verilmektedir.

Tablo 8. Kumaresh ve Baskaran Araştımasında Bulunan Hata Önleme Yöntemleri Uygulandıktan Sonra Proje Hata Değerleri

Proje numarası	**Proje Adı & Tanımı**	**KLOC**	**Hata Sayısı**	**Efor (saat)**	**Hata Yoğunluğu**
Proje 1	eCampus HR Module	39	118	2900	0.003
Proje 2	Content Management System - Additional Reporting	11	54	925	0.005
Proje 3	MOSS based document management system	16	152	1950	0.010
Proje 4	UAE Finance module development	9	97	1260	0.011
Proje 5	R&D based BI tool	33	267	2450	0.008

Araştırma sonucunda, DHS yöntemi kullanılarak sınıflandırılmış hatalar incelenmiştir. Hata tipleri ile yapılan kök neden analizi sonucunda, bulunan hataların giderilmesi için alınan önlemler sayesinde hataların sayısı azaltılarak müşteriye teslim edilen ürün kalitesi arttırılmıştır [20].

Kumaresh ve Baskaran'ın çalışmasının Bölüm 4'te sunulan çalışmadan farkı, hataların sınıflandırılırken dikey sınıflandırma yönteminin birebir uygulanması ve süreç iyileştirme yapılma durumunun hata yoğunluğu değerleriyle kontrolüdür.

3.3. Jalote ve Agrawal'ın Çalışması

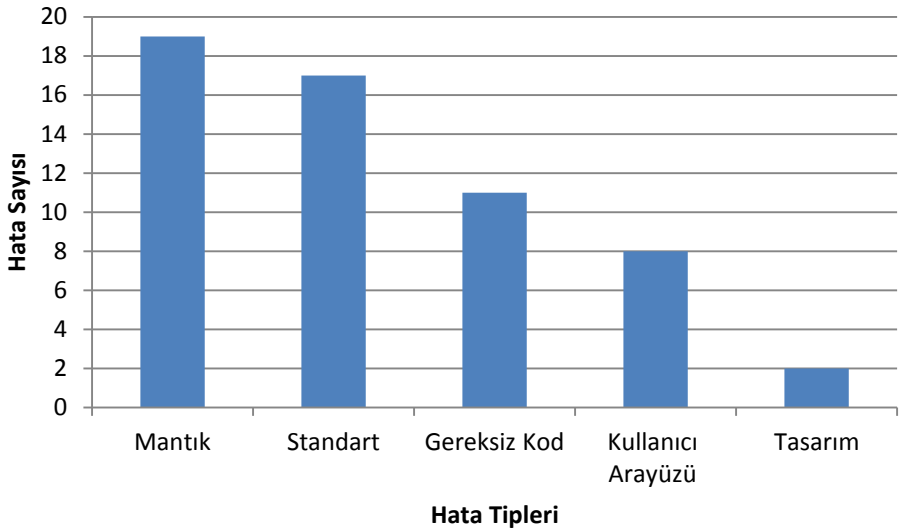
Jalote ve Agrawal tarafından yapılan araştırmada, yazılım sektöründe bulunan hataları azaltma yöntemleri ve tekrarlanan yazılım projelerinde daha verimli ürünler ortaya çıkarmak için hangi teknikler kullanılabileceği kısaca özetlemiştir [34]. Bunun yanında örnek bir çalışma verisi de analizde yer almaktadır.

Analiz kapsamında Rational Unified Process (RUP) terminolojisi uygulanmıştır. Süreç boyunca tespit edilen 57 hatanın hata tiplerine göre dağılımları ve kurulan ilk yapıda bulunan veriler Tablo 9'da verilmektedir.

Tablo 9. Jalote ve Agrawal Araştırmasında İlk İterasyon için Hata Verisi

Hata Tipi	**Mantık**	**Standart**	**Gereksiz Kod**	**Kullanıcı Arayüzü**	**Mimari**
Hata Sayısı	19	17	11	8	2

Bulunan hata türleri mantık, standart, gereksiz kod, kullanıcı arayüzü ve tasarım ana maddelerine ayrılmıştır. Şekil 11'de ise bu verilerin pareto grafiği ile gösterilmiş hali mevcuttur.



Şekil 11.Jalote ve Agrawal Araştırmasında Bulunan Hatalar için Pareto Grafiği

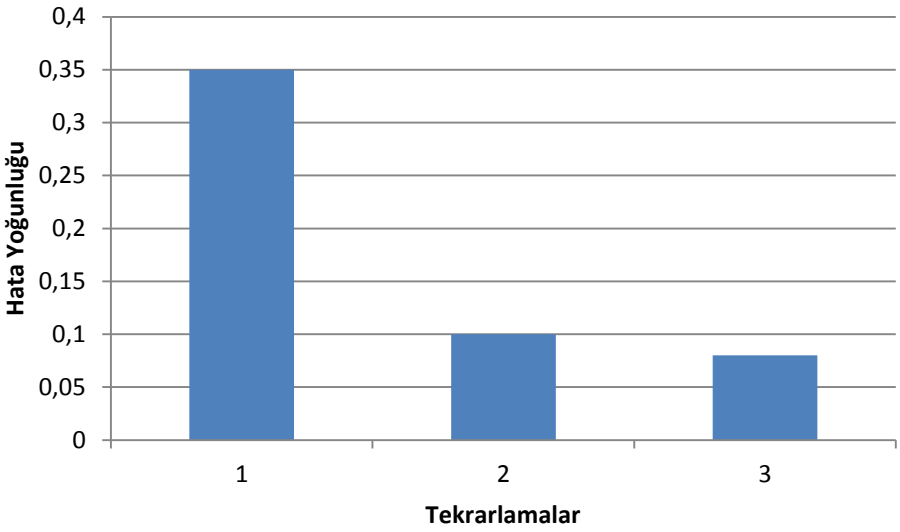
Çalışmanın devamında, en fazla hatanın bulunduğu 3 kategori incelenmiştir. Bu alanlar mantık, standart ve gereksiz koddur. Bulunan bu 3 kategori ile yapılan beyin fırtınası sonucunda kök nedenler ve bu nedenleri önlemek için alınan yöntemler belirlenmiştir.

Hata tiplerine göre alınan önlemler aşağıda verilmektedir;

Standart: Yazılım mühendisleriyle beraber kullanılan standartlar üzerinden geçilecektir

Gereksiz Kod: Kod yazımı ve kod okuma ile ilgili eğitimler verilecektir.

Mantık: Gereksinimlere uygun nasıl kod yazılacağına dair kod yazımı ve kod okuma ile ilgili eğitimler verilecektir.



Şekil 12. Jalote ve Agrawal Araştırmasında Bulunan Değişik İterasyon Sonuçları

Alınan bu önlemler uygulandıktan sonra 3 farklı tekrarlamayla veriler tekrar takip edilmiş ve ölçülmüştür. Şekil 12'de görüldüğü gibi bir saatte yapılan hata sayısı 0.33'den 0.10'a indirilmiştir. Dolayısıyla süreçte kod satır sayısı başına düşen hata sayısının azalması beklentisi sağlanmıştır [34].

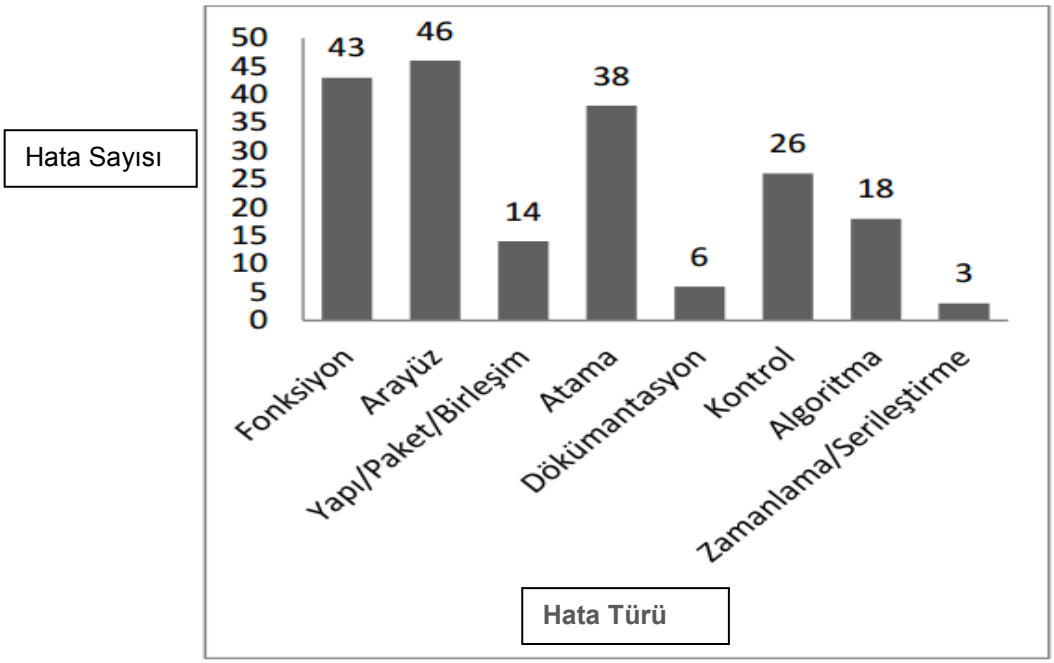
Jalote ve Agrawal araştırmasının Bölüm 4'te bulunan çalışmadan farkı; hataların sınıflandırılması ve analizinde RUP (Rational Unified Process) yöntemi uygulanması ve süreç iyileştirme yapılma durumunun saatte yapılan hata değerleriyle kontrolüdür.

3.4. Söylemez, Tarhan ve Dikici'nin Çalışması

T.C. Aile ve Sosyal Politikalar Bakanlığı Sosyal Yardımlar Genel Müdürlüğü (SYGM) ve TÜBİTAK işbirliği ile başlatılan Bütünleşik Sosyal Yardım Hizmetleri Yazılım Projesi kapsamında 2012 yılında yapılan bir araştırmada kayıt altına alınmış olan toplam 4372 hata incelenmiştir [35]. Hataların tespit edildiği sistemde yazılımın kullanımı ve geliştirilmesi paralel bir ortamda yapılmaktadır. Dolayısıyla analiz kapsamında toplanan veriler ana ve ara sürümler olarak adlandırılan dönemlerde alınmıştır ve her bir ana sürüm ve o ana sürüme ait ara sürümlerdeki hata sayılarını kapsamaktadır. Çalışmada sürümler ilerledikçe hata sayısının değişkenlik gösterdiği, sürüm VO-V3.1 noktasında aniden bir artış olduğu, bir

sonraki sürümde düşüş yaşansa da, daha sonraki sürümde hata sayısının tekrar arttığı belirtilmektedir.

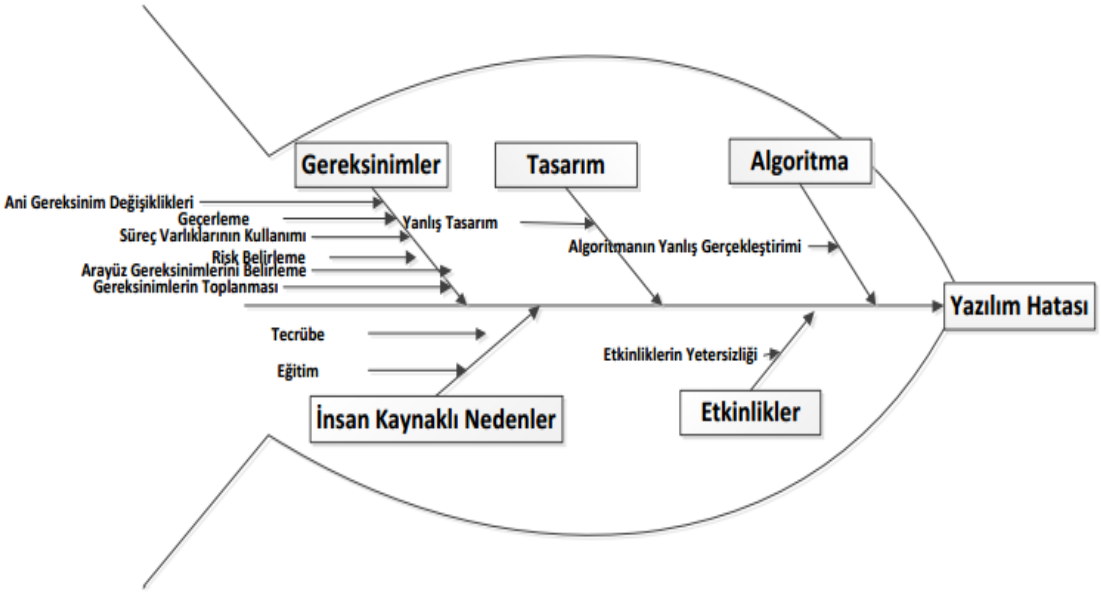
Alınan 220 adetlik hata kümesi üzerinden hata türleri belirlenmeye çalışılmıştır. Hata türleri belirlenirken her hatanın üzerinden teker teker geçilmiş ve ilgili olduğu hata türü belirlenmiştir. Her hatanın türünü belirlemek için ortalama 5 dakika harcanmıştır. Belirlenen hata türlerinin, DHS hata türleri ile örtüştüğü gözlemlenmiştir. 220 adetlik hata kümesinden 194 tanesinin hata türü DHS ile belirlenmiş, geriye kalan 26 tanesinin ise hata türüne tam karar verilememiştir. Bunun sebebi, geriye yönelik bir çalışma yaparak hataların türüne araştırma sırasında karar verilmesi ve DHS hata türleri ile ilgili hataların sınıflandırılamamasıdır. Şekil 13'te hata türlerine göre hata sayılarının dağılımı verilmektedir.



Şekil 13. Söylemez, Tarhan ve Dikici'nin Çalışmasında Bulunan Hata Türü – Hata Sayısı

Daha sonra, Süreç Tutarlılık Matrisi oluşturulmuştur. Sürecin önceden belirlenen ve gerçekleşmesi beklenen girdi, çıktı ve etkinlikleri, bu matrisin satırlarında yer alır. Sütunlardaki değerler ise sürecin her uygulamasında, bu girdi, çıktı ve etkinliklerin gerçekleşme durumunu gösterir. Böylece matris genel olarak, sürecin uygulamada aksayan yönleri hakkında bağlama yönelik bilgi sağlar. Süreç tutarlılık

matrisinden süreç uygulamalarına ilişkin bağlam bilgisi ve DHS ile hataların sınıflandırılması sonucu elde edilen bulgular birlikte değerlendirilerek sebep-sonuç diyagramı oluşturulmuştur (Şekil 14).



Şekil 14. Söylemez, Tarhan ve Dikici'nin Çalışmasında Bulunan Sebep-Sonuç Diyagramı

Şekil 14 yazılım hatasına etki eden faktörlerin ifade edilmesini sağlayarak aslında süreçte nelere dikkat edilmesi gerekiyor, neler hatalı gidiyor, vb. konular hakkında bilgi vermektedir. Bu açıdan nedenlerin tümünü bir arada görmek için sebep-sonuç diyagramının oluşturulması araştırmalar için yararlıdır.

Çalışma sonucunda alınan kararlar ve önlemler aşağıda verilmektedir;

- Versiyonlarda kayıt altına alınan hata sayılarına bakıldığında, hata sayılarında dengesizlik olduğu, bir sonraki sürümde bir öncekinden daha fazla sayıda hataların olabildiği gözlenmiştir. Bu durum DHS tekniğinin önüne geçebileceği bir durumdur.
- Projeden alınan hata verisi kümesi üzerinde yapılan hata türü belirleme işlemi sonucunda DHS hata türlerinin projedeki hatalarla örtüşebildiği, fakat yeni hata türlerinin de oluşabileceği gözlenmiştir. Bu sebeple, kısıtlı bir süre için, yeni hata türlerini belirlemek amacıyla hata bilgisine,

geliştiricilerin hata türlerini kendilerinin girebileceği bir alan eklenerek DHS hata türlerine ek olarak yeni hata türleri belirlenmeye çalışılacaktır.

- Hataları ortaya çıkaran (hata tetikleyici) etkinliklerin ihmal edildiği gözlenmiştir. DHS kullanılması ile etkinliklerin önemi artacaktır. Kod gözden geçirme ve inceleme faaliyetlerinin düzenli bir şekilde yapılmasıyla hataların daha erken tespit edilmesi sağlanabilir [35].

Söylemez, Tarhan ve Dikici'nin çalışmasında yapılan çalışmanın Bölüm 4'te yapılan çalışmadan farkı; hataları sınıflandırma aşamasında DHS yöntemi kullanılmasıdır. Bu araştırmada da Bölüm 4'te yapılan araştırma gibi belirlenen kök sebeplerin iyileştirilmesi henüz sürece uygulanamamıştır.

3.5. Leszak, Perry ve Stoll'un Çalışması

Leszak, Perry ve Stoll tarafından 2000 yılında yapılan bir çalışmada [36] hata analizi ve hataları belirleyen, hataların altında yatan kök sebepleri belirleme üzerinde durulmuştur.

Kök sebep analizi yapabilmek adına internet üzerinde çalışan çevrimiçi modifikasyon isteklerinin toplandığı bir araç kullanılmıştır. Bu aracın amacı; kişilerin verilere daha rahat erişimini ve rahat veri toplanmasını sağlamaktır. Toplanan veriler Modifikasyon İstekleri (MR, Modification Request) veri tabanında toplanmaktadır.

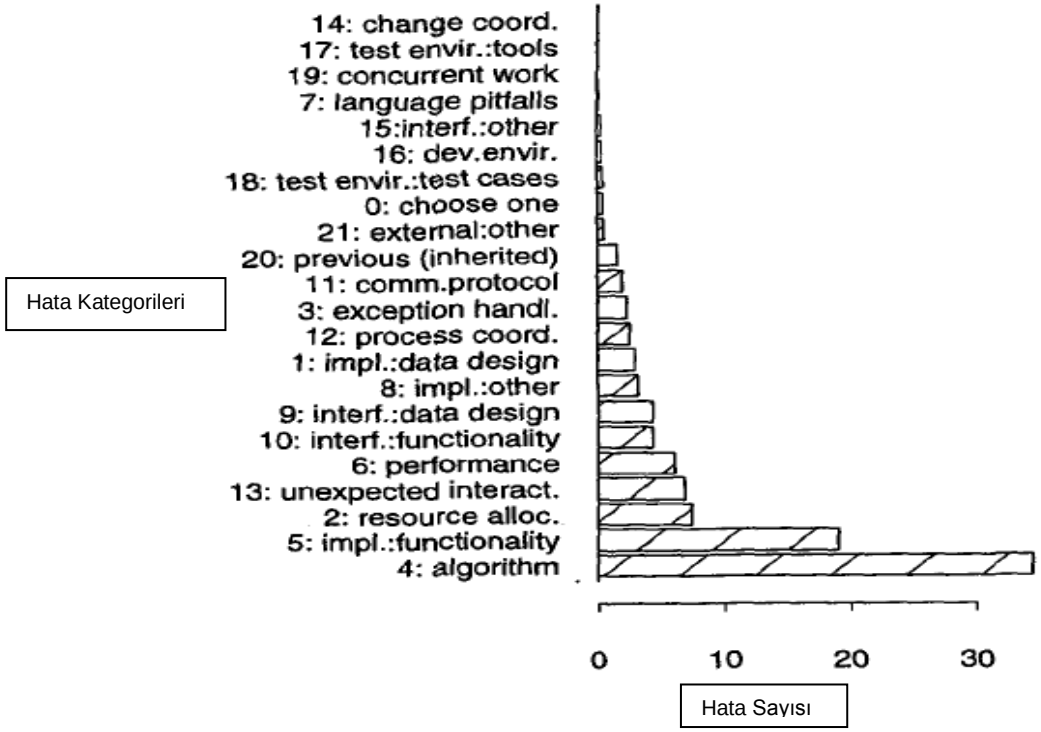
MR verisi toplanırken hatanın bulunduğu faza ait 10 farklı alan bulunmaktadır. Tespit edilen hatalar 3 farklı sınıfta toplanmıştır. Bu alanlar uygulama, arayüz ve harici olarak tanımlanmıştır.

Tablo 10. Leszak, Perry ve Stoll Çalışmasında Bulunan Hata Tiplerinin Sınıflandırılması

Uygulamalar	**Arayüz**	**Harici**
1: veri tasarımı/kullanım	9: veri tasarımı/kullanım	16:çevre kurulumu
2: eleman dağıtımı/kullanım	10: fonksiyonellik tasarımı/ kullanım	17: test çevresi/ araç
3: istisna yönetimi	11: iletişim protokolü	18: test çevresi
4: algoritma	12: süreç koordinasyonu	19: eş zamanlı iş
5: fonksiyonellik	13: beklenmeyen etkileşimler	20: bir sonraki versiyona kalıtım
6: performans	14: değişim koordinasyonu	21: diğer
7: dil güçlüğü	15: diğer	
8: diğer		

Hata sınıflandırmasında bulunan bir üçüncü sınıflandırma alanı ise hatanın niteliği ile ilgilidir. Burada uygulama, arayüz ve harici alanları mevcuttur. Bu alanların tanımlanmasının sebebi verilerin daha belirgin elde edilmek istenmesidir.

Çalışma toplanan 427 veri için yapılmıştır. Toplanan bu hatalar 13 alt sınıfa dağılmıştır. Verilerin dağılımı Şekil 15'te verilmektedir.



Şekil 15. Leszak, Perry ve Stoll Çalışmasında Bulunan Hata Dağılımı

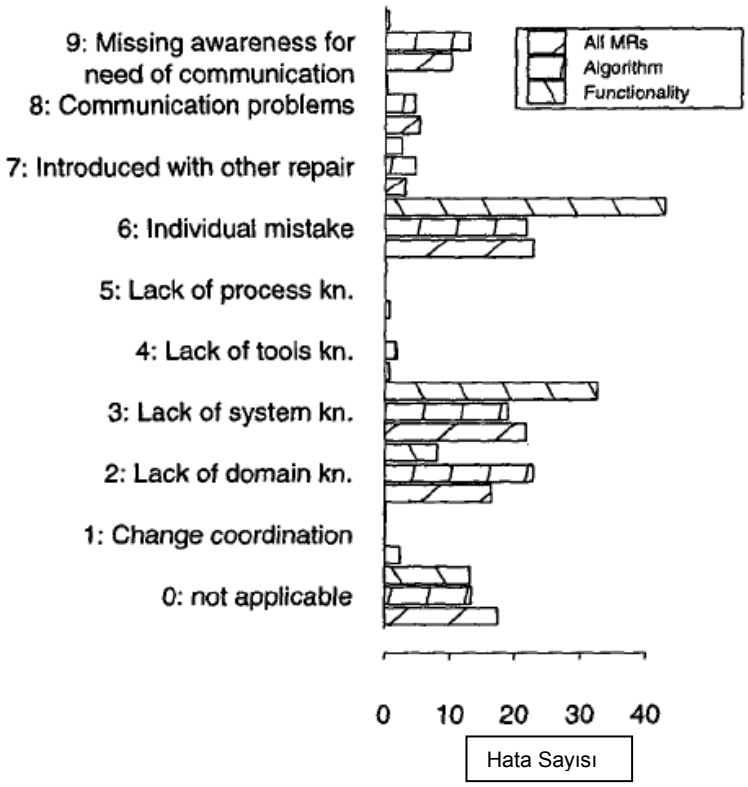
Şekil 15'te verilen Pareto grafiğine göre hatalar en çok algoritma ve fonksiyonellik sınıflarında toplanmaktadır. Bulunan bu 2 sınıflandırmanın, sebeplerine ve tüm hata sayısına olan dağılım Tablo 11'de verilmektedir.

Tablo 11. Leszak, Perry ve Stoll Çalışmasında Bulunan Algoritma ve Fonksiyonellik Hatalarının Fazlara göre Dağılımı

Hata Tipi	**Mimari %**	**Yüksek Seviye Tasarım %**	**Bileşen Teknik Özellik Tasarım %**	**Bileşen Uygulamaları %**
All MRs	4	8	31	40
Algoritma	0	0	48	46
Fonsiyonellik	4	15	25	44

Tablo 11'den de görüldüğü gibi yapılan algoritma hatalarının neredeyse hepsi tasarım aşamasında oluşmaktadır. Fonksiyonellik hataları ise her aşamaya ortalama olarak dağılmaktadır.

Bulunan algoritma ve fonksiyonellik hatalarının belirlenen sebeplere dağılımı ise Şekil 16'da verilmektedir.



Şekil 16. Leszak, Perry ve Stoll Çalışmasında Bulunan Algoritma ve Fonksiyonellik Hatalarının Sebeplere göre Dağılımı

Şekil 16'dan da görüldüğü gibi algoritma hatalarının çoğunun sebebi sistem ve sahanın eksik olmasından kaynaklanmaktadır. Fonksiyonellik hatalarında ise farklı bir biçimde bu değerler az bulunmaktadır. Yetersiz kontrollerden kaynaklanan %84'lük dilimin, %75'i algoritma hatalarından kaynaklanmaktadır. %30'luk kısmı ise fonksiyonellikten kaynaklanmaktadır. Bu çalışma sonucunda ise algoritma ve fonksiyonellik için ön kontrollere daha çok önem verilmesi sonucu ortaya çıkmıştır.

Leszak, Perry ve Stoll'un çalışması, Bölüm 4'te detaylı olarak verilen örnek olay incelemesine en yakın olanıdır. Çünkü bu çalışmada da veriler şirket ağı üzerinde bulunan bir veri tabanında toplanarak kendi içlerinde belirlenen bir sınıflandırma yöntemi ile gruplara ayrılmıştır. Bir farklılık ise Lerzak, Perry ve Stoll'un çalışmasında sebep-sonuç diyagramının kullanılmamış olmasıdır.

4. ÖRNEK VAKA İNCELEMESİ

4.1. Örnek Vaka İncelemesi Arka Planı

Bu örnek vaka çalışmasında Türkiye'de faaliyet gösteren orta ölçekli bir yazılım şirketinin 2006-2012 yılları arasında toplamış olduğu yazılım projelerine ait hata verileri kullanılmıştır. Şirketin isteği doğrultusunda, bu çalışmada şirket ismi verilmeyecektir. Ek olarak şirket içinde kullanılan jargon nedeniyle, kullanılan terimlerin Türkçe karşılıkları verilmesine rağmen İngilizce karşılıklarıyla kullanılacaktır. Söz konusu şirket tarafından yapılan projeler veri gizliliği kapsamında yapılmaktadır. Bu projelerde, Milli Savunma Bakanlığı tarafından projelerin gizlilik durumlarına göre verilen Milli veya NATO kleransına sahip insanlar çalıştırılmaktadır. Şirket kapsamında yapılan projeler temel olarak proje özelliklerine göre 2 farklı gruba ayrılmaktadır. Gizlilik nedeni ile analiz kapsamında bu projeler Tip-1 proje ve Tip-2 proje olarak tanımlanmaktadır.

Tip-1 Projeler, daha sıkı kuralları olan, daha kontrollü kodlama yapılması gereken projelerdir. Ayrıca havacılık yazılımlarında bulunan yazılım sistemleri ve araç sertifikasyonu standartlarına uygun olarak ilerlemesi gerekmektedir. Kullanılmakta olan standartlardan biri DO178-B standartıdır. Yüceer [37], DO178-B kapsamında hataları kısaca şu şekilde tanımlamaktadır.

> "DO-178B yazılımları emniyet değerlendirme kriterlerine göre 5 seviyede değerlendirilir ve bu seviyeler yapılacak aktivitelerin detay ve miktarlarını belirler. Emniyet değerlendirmeleri sonucunda hata durumları seviyeleri aşağıdaki gibi belirlenmiştir.
>
> **Ölümcül = Catastrophic:** Emniyetli uçuş ve inişi engelleyen ve ölümle sonuçlanan hata durumları.
>
> **Tehlikeli = Hazardous/Severe-Major:** Emniyet marjinlerinin veya fonksiyonel yeteneklerin çok büyük oranda düşmesi, fiziksel tehlikenin mürettebatın görev yapmasını engellemesi ve oluşan ters durumların uçaktakilere ciddi yaralanma ve/veya ölüm riski yaratması olarak tanımlanabilecek ve uçağın veya mürettebatın yeteneklerinin kısıtlandığı hata durumları.
>
> **Önemli = Majör:** Emniyet marjinlerinin veya fonksiyonel yeteneklerin büyük oranda düşmesi, fiziksel tehlikenin mürettebatın görev yapmasını kısmen engellemesi ve oluşan ters durumların uçaktakilere ciddi rahatsızlık ve/veya yaralanma riski yaratması olarak tanımlanabilecek ve uçağın veya mürettebatın yeteneklerinin kısıtlandığı hata durumları.

Az Önemli = Minor: Uçağın emniyetini büyük oranda tehlikeye sokmayan ve mürettebatın da yeteneklerinin sınırları dahilinde çalışabildikleri hata durumları.

Etkisiz = No Effect: Uçağın veya mürettebatın operasyonel yeteneklerini etkilemeyen hata durumları.

Yazılım seviyeleri ise bu hata durumlarını oluşturma olasılıklarına göre belirlenir. Aşağıda yazılım seviyeleri ve açıklamaları verilmiştir.

Seviye A: Yazılımın düzgün çalışmaması "Ölümcül" bir hataya yol açabilir veya yol açmasına katkıda bulunabilir.

Seviye B: Yazılımın düzgün çalışmaması "Tehlikeli" bir hataya yol açabilir veya yol açmasına katkıda bulunabilir.

Seviye C: Yazılımın düzgün çalışmaması "Önemli" bir hataya yol açabilir veya yol açmasına katkıda bulunabilir.

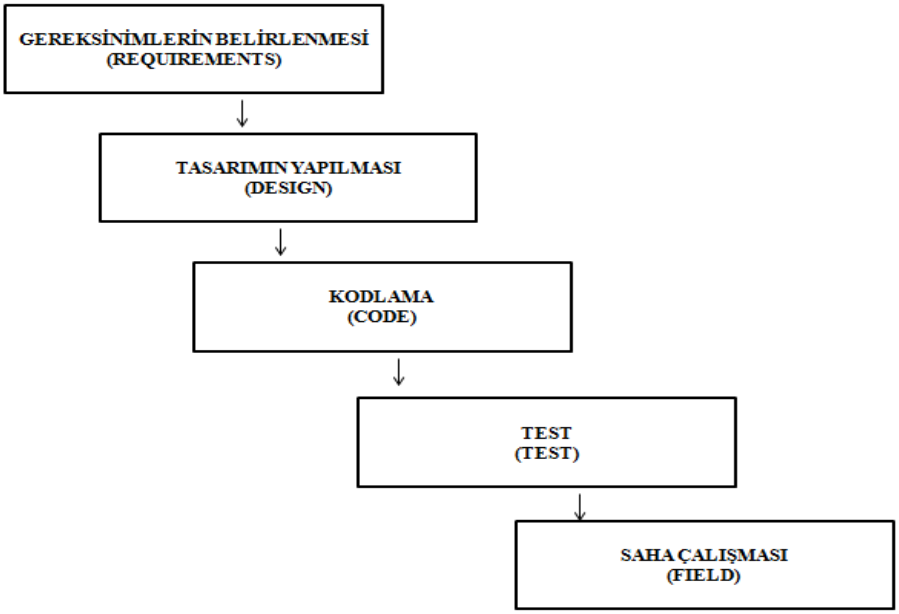
Seviye D: Yazılımın düzgün çalışmaması "Az önemli" bir hataya yol açabilir veya yol açmasına katkıda bulunabilir.

Seviye E: Yazılımın düzgün çalışmaması "Etkisiz" bir hataya yol açabilir veya yol açmasına katkıda bulunabilir [37]."

Tip-2 Projeler ise, sürecin ve kodlamanın daha rahat olduğu, Tip-1 projedeki gibi ön tanımlı ve uyulması zorunlu olan standartların bulunmadığı projelerdir.

Analiz kapsamında toplam 13 proje verisi kullanılmıştır. Kullanılan bu 13 projenin 8 tanesi Tip-1 Proje, 5 tanesi Tip-2 Proje'dir. Analiz kapsamına alınan 13 projede yaşam döngüsü olarak ***Şelale Modeli*** *(waterfall model)* yaşam döngüsünü kullanmaktadır.

Yazılım geliştirmede çok sayıda farklı model ve süreç kullanılmaktadır. Bununla birlikte; yazılım mühendisliğindeki diğer modellere temel teşkil eden Şelale Modeli yazılım yaşam döngüsünü analiz, tasarım, kodlama, test ve bakım olmak üzere beş aşamada ele almaktadır [38]. Şekil 17'de Şelale Modeli yapısı verilmektedir.



Şekil 17. Şelale Modeli

Şelale Modeli yaşam döngüsü kapsamında bir faz tamamlanmadan bir diğer faz başlayamamaktadır. Fazların sırası genellikle "Gereksinimlerin belirlenmesi" (Requirements), gereksinimlere uygun "Tasarımın yapılması (Design)", tasarıma uygun "Kodlamanın yapılması" (Code), yazılan kodun doğruluğuna yönelik "Testlerin yapılması" (Test) ve son olarak ürünün müşteriye teslim edilmesi ve "Saha çalışmalarının yapılması" (Field) olarak özetlenebilir. Yapılan projenin gereksinimleri çok net ve proje boyunca değişmeyecekse, yaşam döngüsü olarak Şelale Modelinin seçilmesi uygun olmaktadır. Aksi bir durumda bu yaşam döngüsünün seçilmiş olması projeye beklenenden çok daha fazla iş yükü getireceğinden maliyetin de artmasına neden olacaktır [4].

Özbilgin ve Özlü'nün [39] anlatımıyla kısaca yukarıda verilmekte olan Şelale Modelinin temel fazlarında ne gibi işlemler yapıldığına değinilmek gerekirse;

Gereksinimleri Belirleme: Yazılımın ne yapacağını ve nasıl yapacağının belirlendiği yani problemin tanımlandığı aşama gereksinimleri belirleme aşamasıdır. Öncelikle yazılımdan ne istendiğinin doğru bir biçimde tanımlanması gerekir. Gereksinimleri Belirleme ve analiz aşaması personel, donanım ve sistem gereksinimlerinin belirlenmesi, sistemin fizibilite çalışmasının yapılması, kullanıcıların gereksinimlerinin analizi, sistemin ne yapıp ne yapmayacağının kısıtlamalar göz önüne alınarak belirlenmesi, bu bilginin kullanıcılar tarafından doğrulanması ve proje planı oluşturulması adımlarından oluşmaktadır.

Tasarım: Belirlenen gereksinimlere yanıt verecek yazılımın temel yapısının oluşturulduğu aşamadır. Yazılım tasarımı, bir bileşen veya sistemin nasıl gerçekleştirileceğini belirlemek için kullanılan teknikler, stratejiler, gösterimler ve örüntüler ile ilgilidir. Bu aşama, yazılım bileşenleri arasındaki içsel ara yüzler, mimari tasarım, veri tasarımı, kullanıcı ara yüzü tasarımı, tasarım araçları ve tasarımın değerlendirilmesi alt süreçlerini de kapsamaktadır. Tasarım aşaması, yazılımın hem kullanıcı ara yüzünü hem de programın omurgasını ortaya koymaktadır. Yapılacak tasarım, yazılımın işlevsel gereksinimlere uygun olmasının yanı sıra kaynaklar, performans ve güvenlik gibi kavramları da göz önüne alınarak gerçekleştirilmelidir.

Kodlama: Kodlama aşaması, tasarım sürecinde ortaya konan veriler doğrultusunda yazılımın gerçekleştirilmesi aşamasıdır. Bu süreç programlama çalışmalarının yanı sıra yazılımın geliştirilmesi ve kullanıcıya ulaştırılması sürecindeki bütün çalışmaları kapsar. Tasarım sonucu üretilen süreç ve veri tabanının fiziksel yapısını içeren fiziksel modelin bilgisayar ortamında çalışan yazılım biçimine dönüştürülmesi çalışması olarak da nitelendirilebilir.

Test: Test aşaması, yazılım kodlanması sürecinin ardından gerçekleştirilen sınama ve doğrulama aşamasıdır. Elde edilen uygulama yazılımının hem belirlenen gereksinimleri sağlayıp sağlamadığı hem de sonucun beklentilere uygun olup olmadığının kontrolü yapılır. Yazılım üretiminde ilk testler genelde geliştirme sürecinde programcı tarafından yapılır. Bununla birlikte, asıl hata ayıklama ve geribildirim hizmeti test ekipleri tarafından yapılır. Testler ve geribildirim müşteri yazılımı kullandığı sürece devam eder. Test sürecinde en faydalı geribildirimler son kullanıcı test gruplarından gelir.

Saha: Yazılımın müşteriye tesliminden sonra hata giderme ve yeni eklentiler yapma aşamasıdır. Yazılımın kullanıma başlanmasından sonra yazılımın desteklenmesi sürecini kapsar. Yazılımın eksiklerinin giderilmesi, iyileştirilmesi gibi alt aşamaları içeren aşamadır [39].

Analiz kapsamında kullanılan veriler şirket tarafından kullanılan bir araç[8] vasıtasıyla bir veri tabanında toplanmaktadır. Proje yaşam döngüsü boyunca toplanan hata kayıtları projede çalışan elemanlar tarafından girilmiştir. Giriş yapılırken kullanılan alanlar ve karşılık değerleri Tablo 12'de verilmektedir.

[8] JIRA yazılımı projeler içerisinde, belirli bir hayat döngüsü olan iş maddelerinin (issue) oldukça gelişmiş bir şekilde takibinin, yönetiminin ve raporlamasının yapılabilmesine olanak sağlayan, kurumsal yapılanmalara göre değiştirilip düzenlenebilen, JAVA platformu üzerinde geliştirilmiş bir araçtır [37].

Tablo 12. Hata Giriş Alanları ve Değerleri

ALAN	DEĞER
Proje Adı (Project Name)	Proje adının veritabanında geçen kısaltması.
Hata Tipi (Defect Type)	Bulunan hatanın tipi aşağıda verilen 3 tipten bir tanesi olmak zorundadır. **Software Change Request (SCR):** hazırlanan doküman veya ürün müşteriye gönderildikten sonra gelen değişiklik ve /veya hata isteği. **Peer Review (PR):** hazırlanan doküman veya ürün müşteriye gönderilmeden önce bulunan değişiklik ve /veya hata isteği. **Informal Action Item (INFAI) :** Ürün geliştirilme esnasında bulunan hata ve /veya değişiklik isteği.
Hata Numarası (Number of Defects)	Bulunan her hata için kullanılan eşsiz sayı.
Hatanın Bulunduğu Faz (When Found)	Hata bulunduğunda işlem yapılan faz Requirements, Design, Code, Test, Field
Hatanın Kaynaklandığı Faz (Defect Origin)	Bulunan hatanın kaynaklandığı faz. Requirements, Design, Code, Test, Field
Hata Sayısı	Bulunan hata sayısı.
Hata Kategorisi (Defect Category)	Bulunan hatanın çeşidine göre gruplara ayrılmış olan kategoriler, DHS yöntemi metodunda kullanılan hata tipleri baz alınarak ve biraz da farklılaştırıllarak kategorilere ayrılmıştır. A,B,C,D,E,F,G,H,I,J,K
Hata Önem Derecesi (Defect Severity)	Bulunan hatanın proje ve ürün üzerinde bütçe ve takvim olarak bulunan etkisi •**Kritik:** Yazılımın veya sistemin çalışmasını durduran problemler. •**Majör:** Projenin yanlış çıktı üretmesini sağlayan problem •**Minör:** Projenin çalışmasını etkileyen, düzeltilmesi kolay problem.

Gizlilik derecesi yüksek ve hakları saklı projeler kullanılmasından kaynaklı şirketin talebi doğrultusunda verilerde çeşitli değişiklikler yapılmıştır. Şirket tarafından belirlenen bir algoritma ile çalışan bir kara kutuya[9] sokulan gerçek verilerden bu çalışma kapsamında kullanılan veriler elde edilmiştir.

Şekil 18. Kara Kutu İşlemi

Dolayısıyla analiz kapsamında kullanılmış olan değerler gerçek veriyi yansıtmamaktadır. Projeler sırasında toplanan hata verilerinin tutulduğu alanların hepsine aynı işlem uygulanarak sayılar farklılaştırılmıştır. Verilere uygulanan formül şirket tarafından belirlenmiştir ve gizlilik nedeni ile bu çalışmada açıklanmamaktadır.

4.2. Örnek Olay İncelemesi Çalışmaları

Analiz kapsamında kullanılan veriler şirket veri tabanından 30 Mayıs 2012 tarihinde alınmıştır. Veriler 2006-2012 yılları arasında tutulan 6 yıllık veriyi kapsamaktadır. Alınan tüm veriler doğruluk, bütünlük ve tutarlılık açısından kontrol edilmiş; hatalı, tutarsız ve eksik alanlar projelerde çalışan kişiler ile beraber düzeltilmiştir. En sık karşılaşılan veri sorununun; hatanın kaynaklandığı faz (defect origin) ve hataların hangi fazda bulunduğu (when found) alanlarının birbirleri ile karıştırılması olduğu gözlemlenmiştir. Sorunlu veriler, verileri giriş yapmış olan kişilerle görüşülerek düzeltilmiştir.

[9] Kara Kutu işleminde veriler belirlenen bir algoritmaya sokularak değiştirilir. Algoritmalar önceden bilinen formüller ve ya algoritmalar olacağı gibi analiz yapan kişiler tarafından belirlenen formüller de olabilmektedir.

4.2.1. Verilerin grafikler ile sunumu

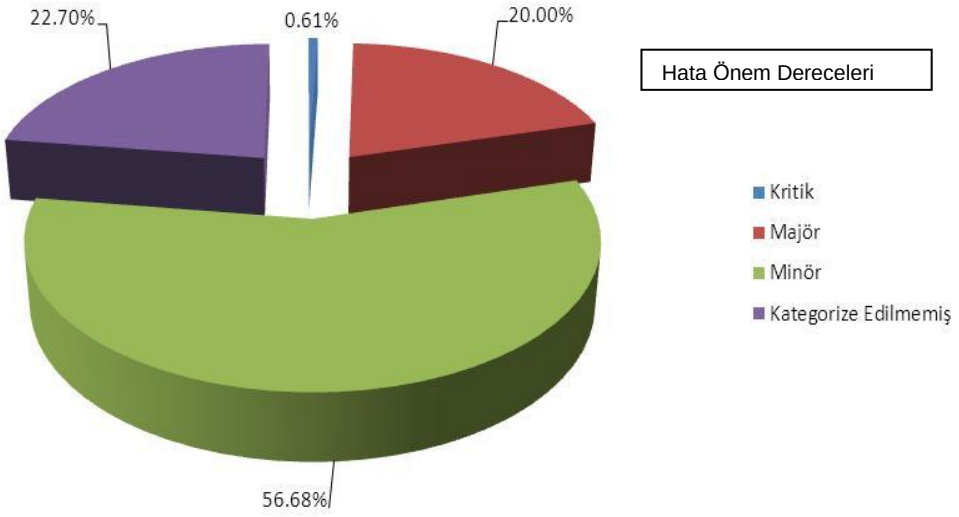
Veri tabanında bulunan hatalar ile ilgili daha rahat fikir yürütülebilmesi, hataların süreç boyunca ne gibi dağılımlar gösterdiğinin daha rahat anlaşılabilmesi için analiz kapsamında toplanan veriler çeşitli grafikler aracılığı ile incelenmiştir.

Tablo 13'te özetlendiği ve Şekil 20'de gösterildiği üzere, ürün müşteriye gönderilmeden önce yapılan ön kontroller (PR+INFAI) ile tüm projelerde bulunan toplam hatanın % 76'sı yakalanmıştır.

Tablo 13. Hata Tipleri ve Önem Dereceleri (Tüm Projeler)

Hata Tipi	*INFAI*	*PR*	*SCR*	*TOPLAM*
Kritik		10	214	***224***
Majör		4050	3235	***7285***
Minör	1650	15304	3691	***20645***
Kategorize Edilmemiş	5360	1546	1364	***8270***
Toplam	***7010***	***20910***	***8504***	***36424***

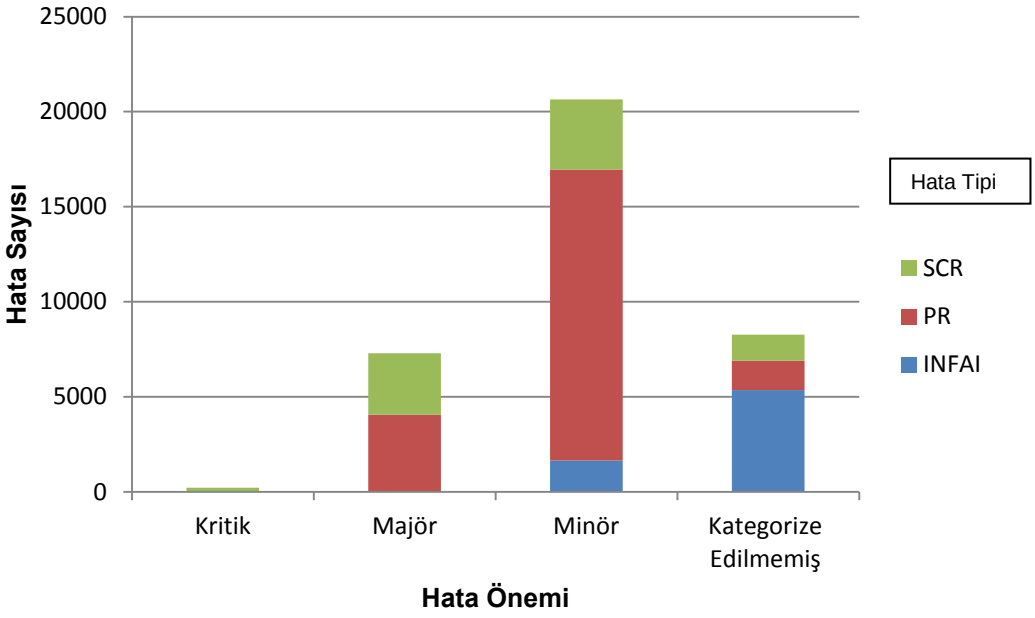
Bulunan bu sonuç, bölüm 2'de bahsedilen yazılım mühendisliği projelerinde dikkat edilmesi gereken 10 kuralın onuncusu tarafından da desteklenmektedir (Birçok yazılım mühendisi verileri takip etmek amaçlı Pareto grafiklerini kullanmaktadır. Bu grafikler %80'lik etkinin %20'lik kısımdan geleceğini söylemektedir).



Şekil 19. Hata Önem Dereceleri

Şekil 19'da verilmekte olan bulunan hataların önem derecelerine bakıldığında, hataların %21'lik bölümü majör ve kritik hatalardan, %56.68'lik bölümü ise minör hatalardan kaynaklanmaktadır. Şekil 19'dan anlaşılacağı üzere, majör ve kritik hataların toplamının azaltılması için çalışmalar yapılması gerektiği ortaya çıkmaktadır.

Hataların önem derecelerine göre hangi tip kontroller esnasında kayıt altına alındığı bilgisi Şekil 20'de verilmektedir.



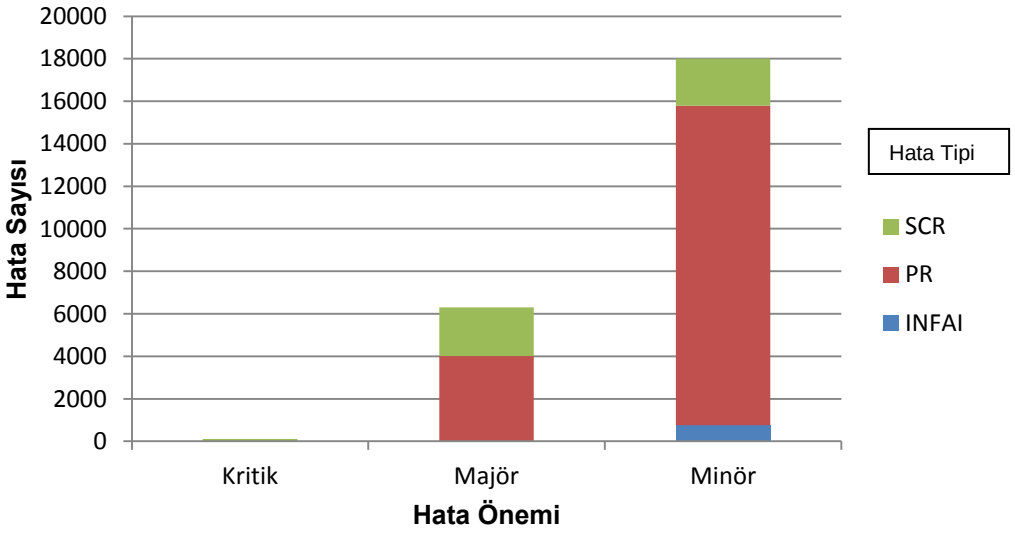
Şekil 20. Hata Önemi ile Hata Tipi (Tüm Projeler)

Şekil 19 ve Şekil 20'de görüldüğü gibi bulunan hataların büyük bir bölümü minör hatadır. Kategorize edilmemiş veri olarak görülen veri grubu eski (2006) bir proje verisini kapsamaktadır. Projenin yürütüldüğü esnada, tutulan alanlarda hatanın önem derecesi olmadığından dolayı bu şekilde kategorize edilmemiş veri olarak görülmektedir. Proje verisinin diğer alanları sorunsuz ve eksiksiz bir şekilde doldurulduğundan, analiz kapsamından çıkartılmamıştır. Kritik hata sayısının diğer gruplar ile kıyaslandığında çok daha az sayıda olması şirket açısından iyi bir durumdur. Fakat dikkat edilmesi gereken nokta, bulunan kritik hataların hepsinin SCR, yani müşteriye giden hata olmasıdır. Bulunan minör hataları incelediğimizde büyük bir bölümü (16,954) PR ve INFAI aşamasında bulunmaktadır. Önceden de değinildiği gibi, bu alanlar, ürün müşteriye gitmeden önce iç kontrollerle düzeltilen

hata sayısını vermektedir. Bu alanların yüksek sayıda veri bulundurması yapılan iç kontrollerin amacına ulaştığının bir göstergesidir.

Şekil 20'de bulunan veri Tip-1 ve Tip-2 proje verisini bir arada bulundurmaktadır. Fakat proje tiplerine göre incelendiğinde proje yapılarının birbirinden farklı olmasından dolayı grafikte değişimler olmaktadır.

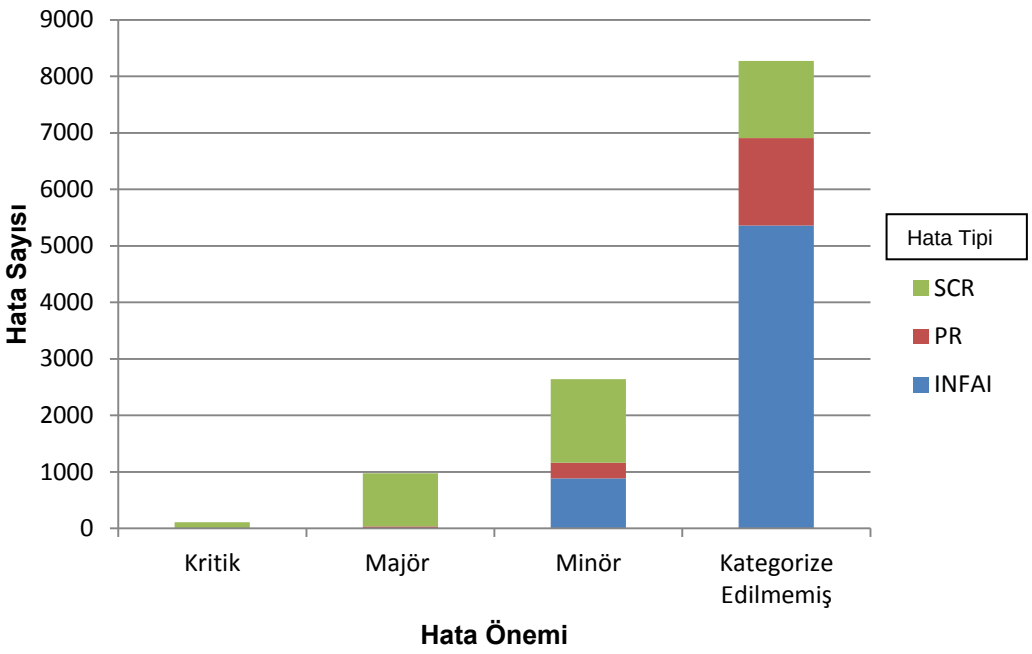
Tip-1 projeler için bulunan veriler tekrar incelendiğinde Şekil 21'de bulunan grafik elde edilmektedir.



Şekil 21. Hata Önemi ile Hata Tipi (Tip-1 Proje)

Tip-1 projelere bakıldığında kategorize edilmemiş veri bulunmamaktadır. Bunun sebebi kategorize edilmemiş verinin Tip-2 Proje verisi olmasıdır. Tip-1 projeler için dağılımda genel yapıya paralel bir görüntü bulunmaktadır. Majör ve minör hatalara bakıldığında büyük bir bölümünün PR verilerinde bulunması ve çözülmesi müşteriye gönderilen ürünün çok daha az hata ile gönderilmesini sağlamaktadır. Fakat SCR hatalarının da minimuma indirilmesi için çalışmalar yapılmalıdır.

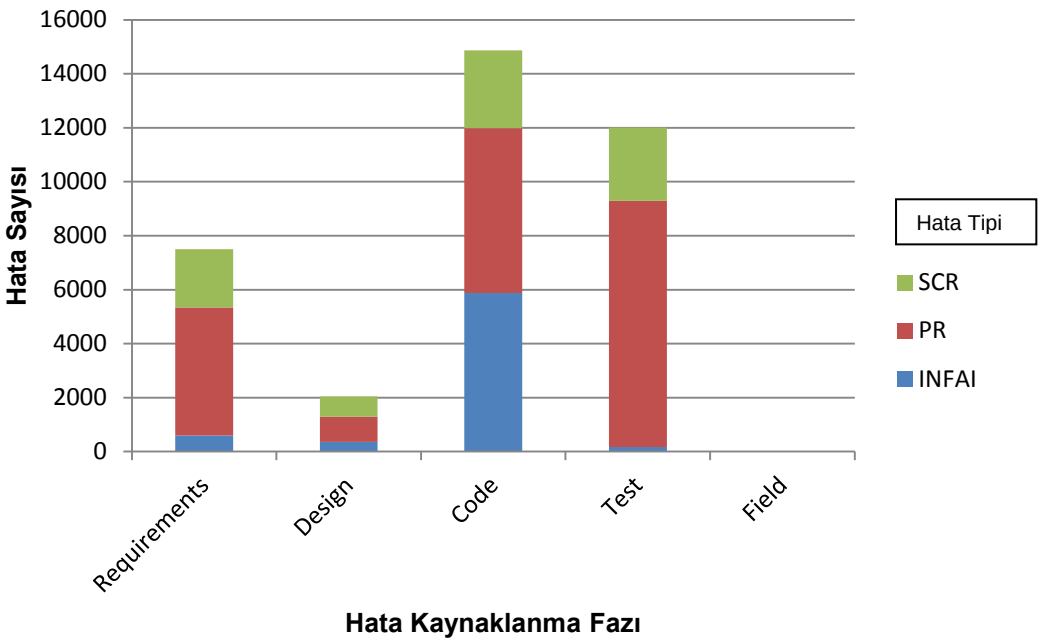
Aynı açıdan Tip-2 projeleri Şekil 22'de verilmektedir.



Şekil 22. Hata Önemi ile Hata Tipi (Tip-2 Proje)

Bulunan hataların önem dereceleri ve hata tiplerinin sadece Tip-2 proje verileriyle incelenmiş hali Şekil 22'de verilmektedir. Tip-2 projelerinde göze çarpan nokta kritik ve majör hata önem derecelerinde bulunan verilerinin büyük bir bölümünün SCR olarak kaydedilmiş olmasıdır. Bu gözleme dayanarak; PR ve INFAI süreçlerinin daha dikkatli yapılması gerektiği ve müşteriye gönderilen üründe bu şekilde önem derecesi yüksek hataların sayılarının azaltılması için çalışmalar yapılması gerektiği tespit edilmiştir.

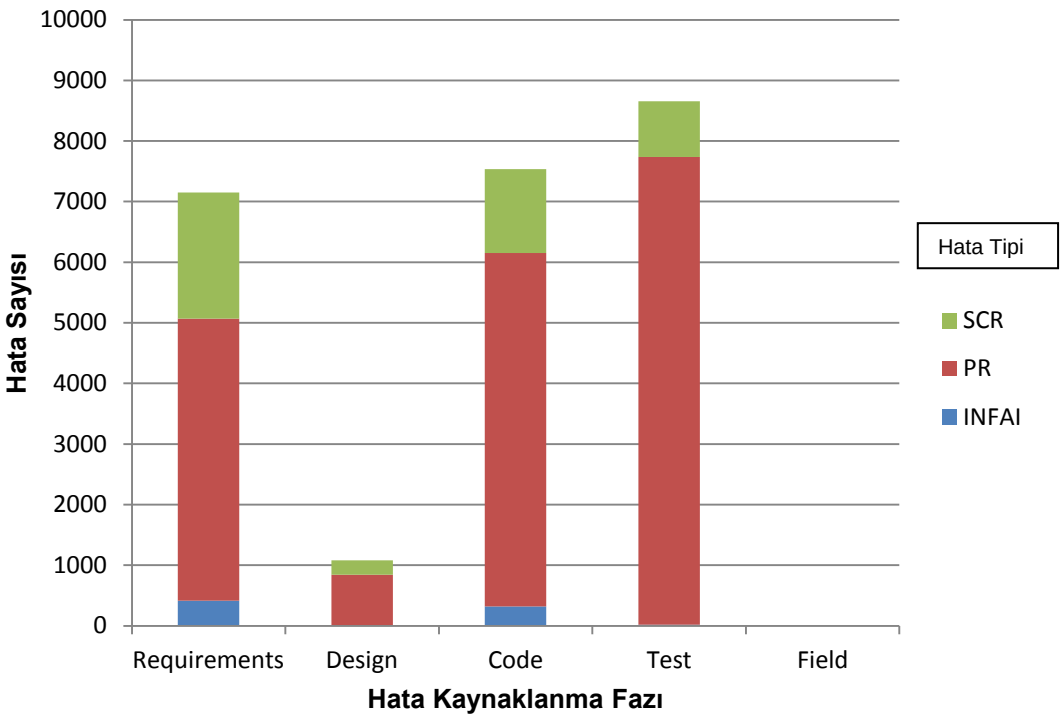
Hataların önem derecelerine göre hangi fazdan kaynaklandığı verisi bir arada incelediğinde elde edilmektedir.



Şekil 23. Hata Kaynaklanma Fazları ile Hata Tipleri (Tüm Projeler)

Şekil 23'te görüldüğü gibi en çok Kodlama (Code) aşamasından kaynaklı hata bulunmuştur. Bu inceleme sayesinde şirket genelinde kodlama aşamasına daha fazla önem verilmesi gerektiği ortaya çıkmaktadır. Kodlama aşamasında yazılan kod, çeşitli kontrollerle gözden geçirilerek bulunan hata sayısı azaltılabilir.

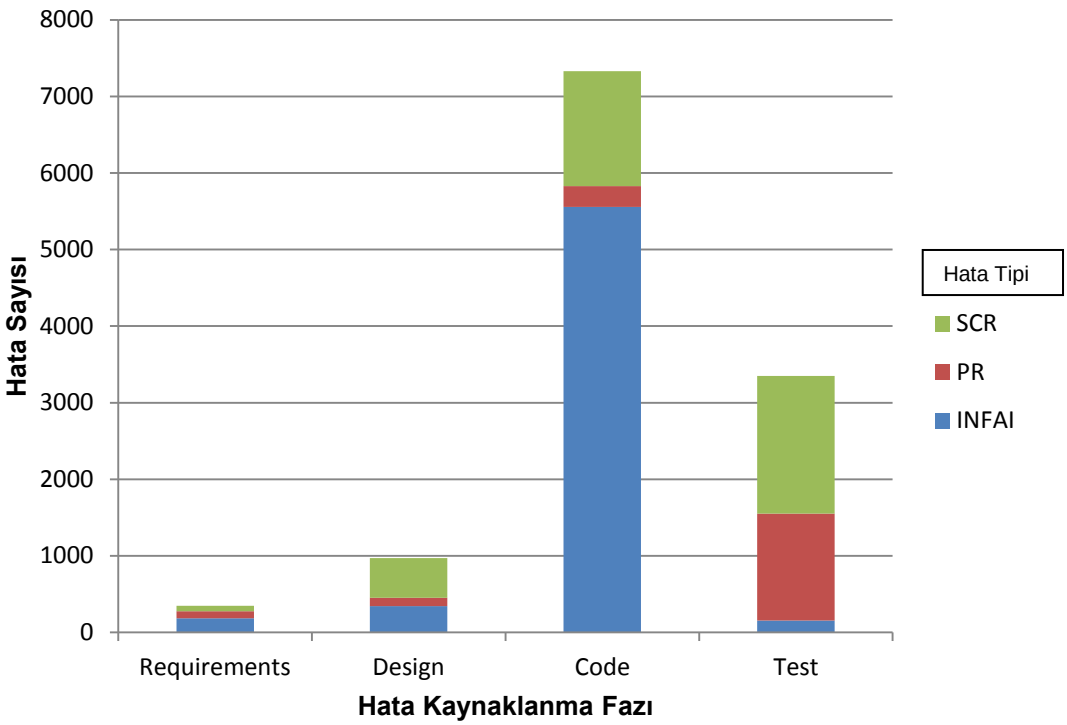
Aynı grafiği Tip-1 proje ve Tip-2 Proje kırılımlarına bakarak incelersek, Tip-1 Proje çeşitleri için aynı kırılım Şekil 24'te verilmektedir.



Şekil 24. Hata Kaynaklanma Fazları ile Hata Tipleri (Tip-1 Proje)

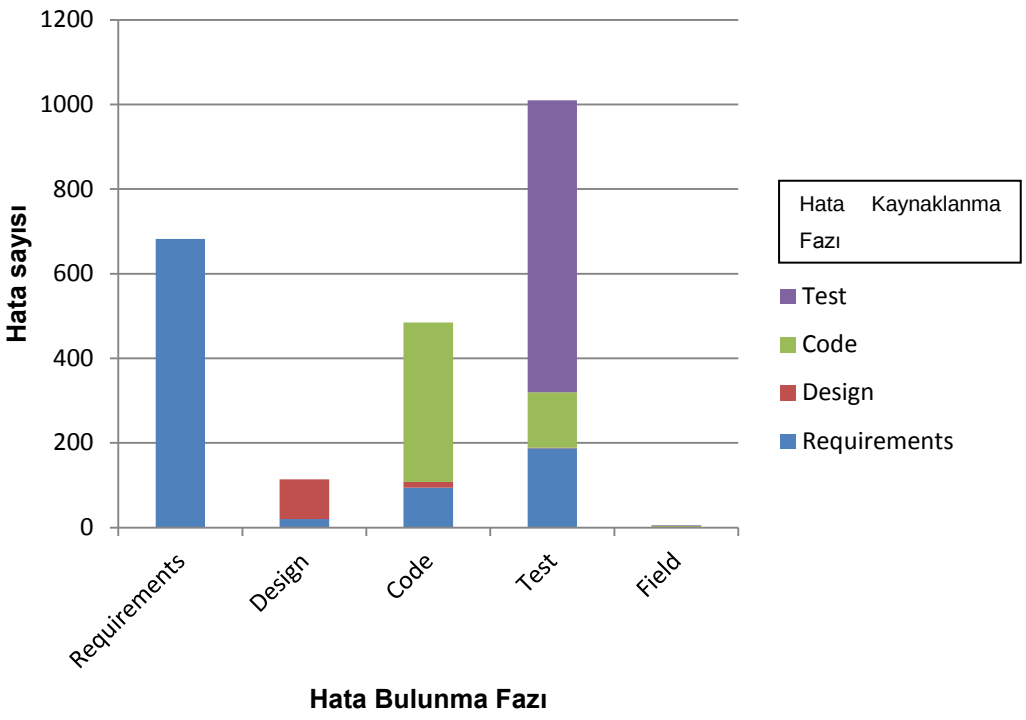
Genel yapıya göre farklılık gösteren Tip-1 Proje verilerinde en çok hatanın kaynaklandığı faz Code aşamasından Test aşamasına kaymaktadır. Requirements aşamasında da Şekil 23'e göre gözle görülür bir artış vardır. Tip-1 proje verilerinde Saha (Field) aşamasından kaynaklı hiç hata olmaması, müşteri tarafından bu evrede sorun bulunmadığının göstergesidir.

Tip-2 Projelerinin dağılımı Şekil 24'te görüldüğü gibi Tip-1 Proje verilerine göre oldukça farklılık göstermektedir. Tip-1 Proje verilerinde Requirements, Code ve Test aşamasından kaynaklanan hatalar oldukça fazlayken Tip-2 projelerinde bu dağılım Code aşamasına yoğunlaşmıştır. Dolayısıyla Tip-2 projelerde kodlama aşamasına daha çok önem verilmeli ve belki de bu evrede çalışan kişiler eğitimlerle bilgilendirilmelidir.



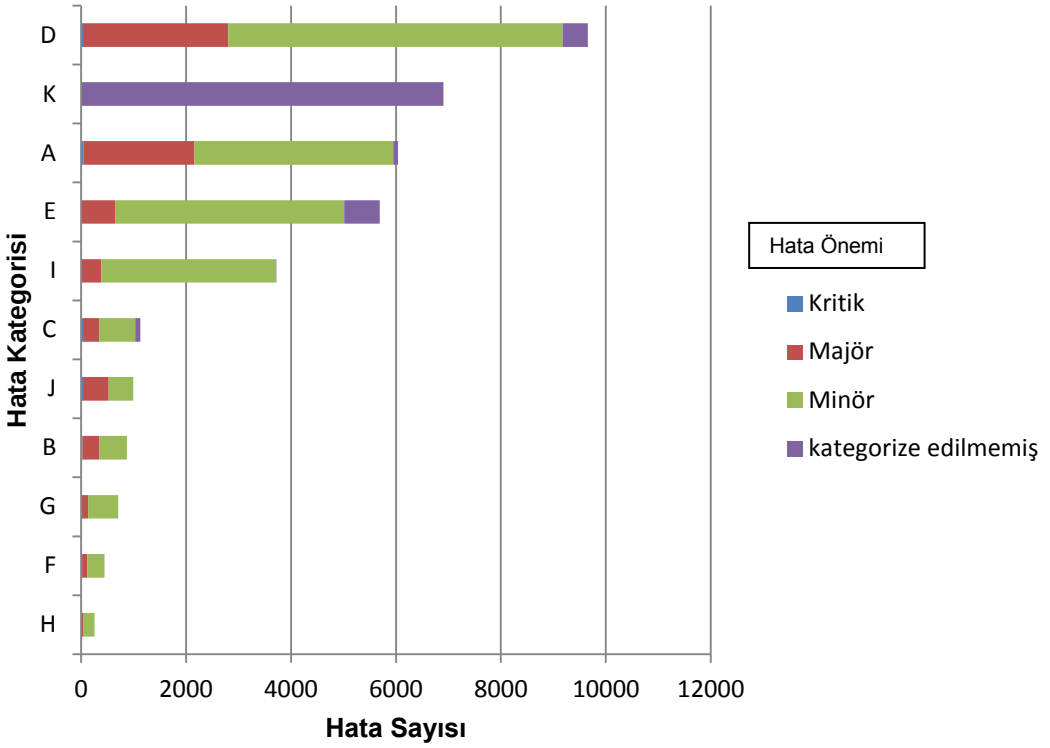
Şekil 25. Hata Kaynaklanma Fazları ile Hata Tipleri (Tip-2 Proje)

Bölüm 4.1'de değinildiği gibi, analiz kapsamında bulunan veriler Şelale Modeli yaşam döngüsüne sahip projelerdir. Bu yaşam döngüsü modelinde bir sonraki aşamaya geçerken bir önceki aşamanın tamamen bitirilmiş olması gerekmektedir. Dolayısı ile Şekil 26'da bulunan grafiğe bakıldığında proje bir sonraki faza doğru ilerledikçe, önceki fazlara ait hata çıkmaması istenmektedir. Şekil 26'da görüldüğü üzere Test fazına gelindiğinde hala Requirements, Design, Code fazlarından kaynaklı hatalar bulunmaktadır. Bu aşamada bulunan Requirements hatası geriye dönük bir gereksinimi etkilediğinden bu değişen gereksinimle ilgili tasarımın, kodlamanın ve testin tekrar yapılması gerekecektir. Dolayısı ile önceki fazlara etki edecek olan her hata projede planlanmayan ekstra efor harcanmasına neden olacaktır. Bölüm 2.1'de bahsedildiği üzere Test aşamasında bulunmuş Requirements fazı kaynaklı bir hata için 100 kat fazla efor ve maliyet harcanmaktadır.



Şekil 26. Hata Bulunma Fazı ve Hata Kaynaklanma Fazı (Tüm Projeler)

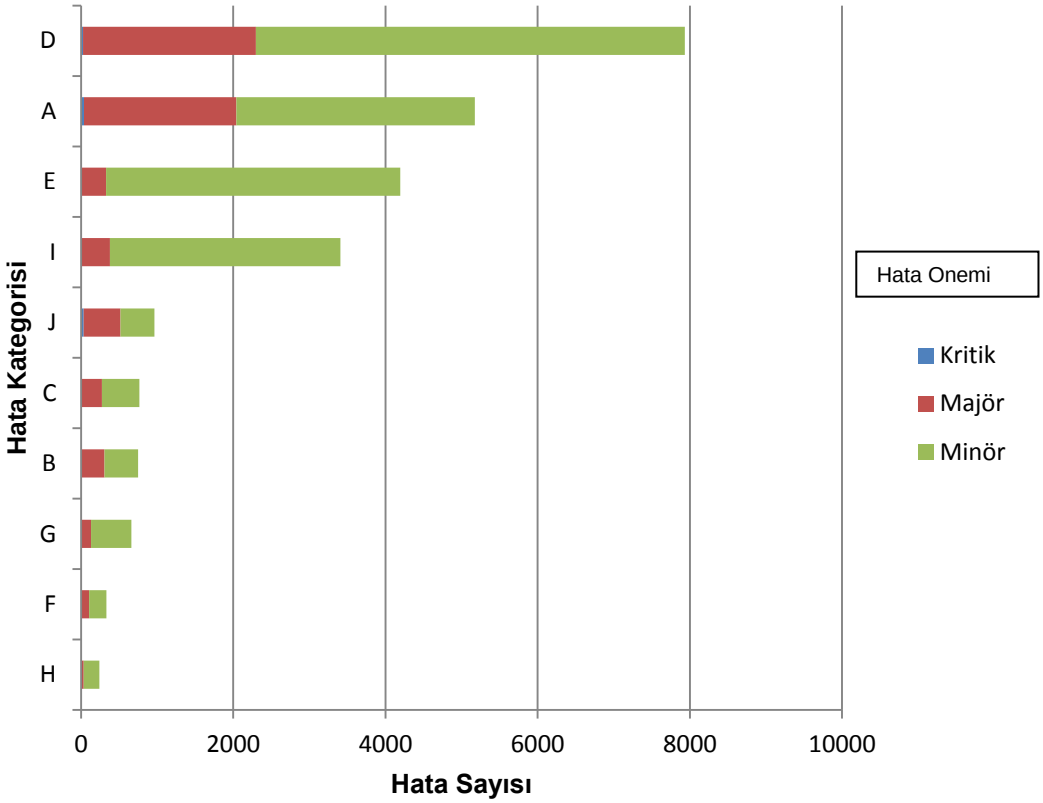
Analiz kapsamında bulunan hatalar problem çeşidine göre belirli kategorilerde toplanmaktadır. Şirketin isteği üzerine bu kategorilerin isimleri ve ayrıntıları açık bir şekilde verilememektedir. Hataların bulunma esnasında veri tabanına girişleri yapılırken 11 adet hata kategorisi bulunmaktadır. Bu kategoriler sayesinde hatanın kök nedenine daha rahat ulaşılabilmektedir. Şirket kapsamında bulunan tüm hataların bu kategorilere göre dağılımı Şekil 27'de verilmektedir.



Şekil 27. Hata Kategorileri ile Hata Önemi (Tüm Projeler)

Toplanan tüm hatalar incelendiğinde D, K ve A kısaltmalarıyla yer alan kategoriler hataların büyük bir bölümünü içermektedir. Fakat K hata tipi kategorize edilmemiş veriyi kapsadığından kök neden analizi kapsamında yer almayacaktır. Kategorize edilmemiş veri kapsamında şirketin 2006 yılında Tip-2 Projelerden topladığı verilerin bir bölümü bulunmaktadır. D kategorisinde bulunan hataların çok büyük bir bölümü minör hatalardan oluşsa da majör hata oranı da bir o kadar fazladır. Bu nedenle D kategorisinde bulunan hata türü kök neden çalışması ile incelenerek, yapılan majör hataların azaltılması için önlem alınmalıdır.

Hata kategorileri verisine proje tipleri için ayrı ayrı bakılırsa; Tip-1 projeler için Şekil 28 elde edilmektedir.

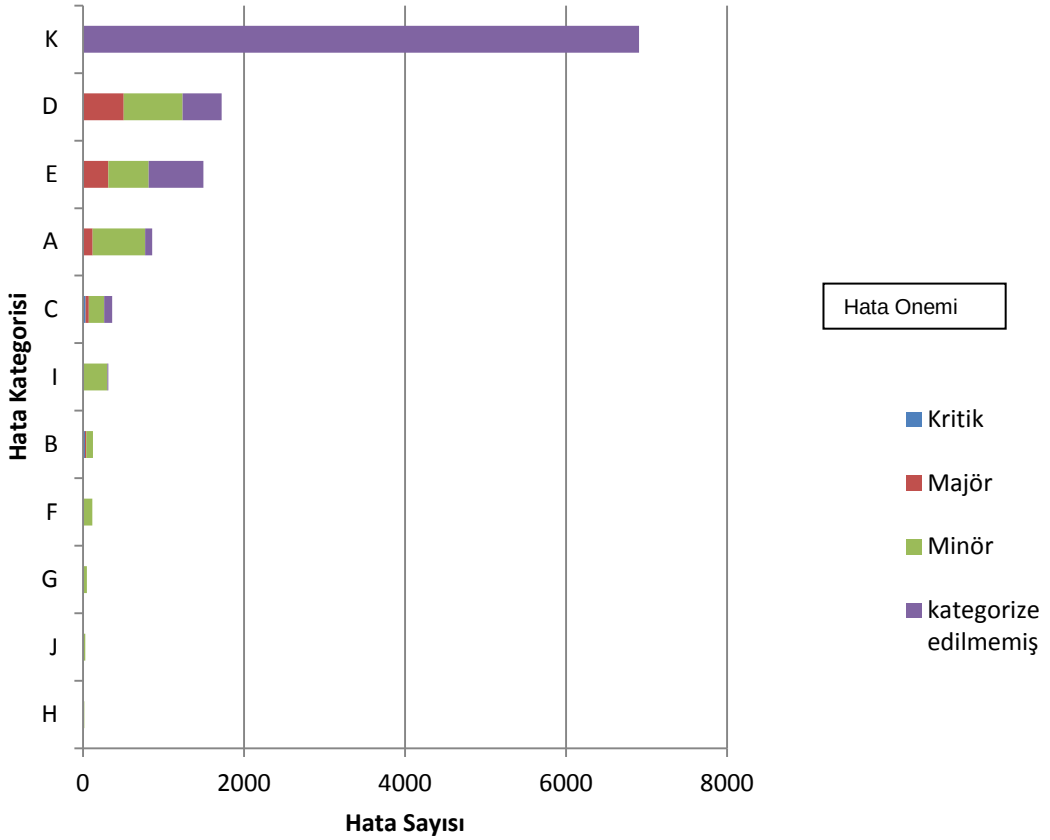


Şekil 28. Hata Kategorileri ile Hata Önemi (Tip-1 Proje)

Tip-1 Proje verilerine bakıldığında hatalar çoğunlukla D, A ve E kategorilerinde toplanmıştır.

D kategorisinde bulunan veriler ile A kategorisinde bulunan veriler, hataların çoğunlukla yapıldığı alanları işaret ettiğinden, bir sonraki aşamada yapılan kök neden analizi kapsamında incelenmiştir.

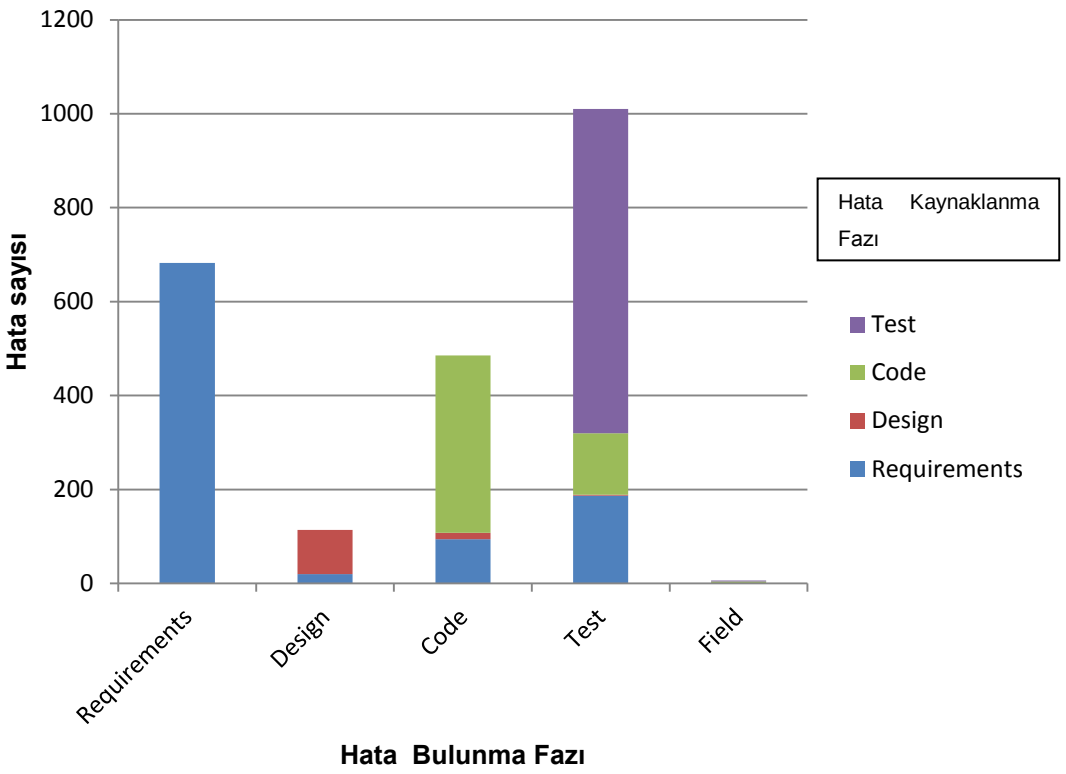
Tip-2 orojeleri için aynı kırılım Şekil 29'da verilmektedir.



Şekil 29. Hata Kategorileri ile Hata Önem Dereceleri (Tip-2 Proje)

Tip-2 Proje çeşitlerine bakıldığında hataların K, D ve E kategorilerinde toplandığı görülmektedir. K kategorizasyonu önceden de bahsedildiği üzere, kategorize edilmemiş verilerden oluştuğundan kök neden analizi kapsamına alınmayacaktır. Dolayısıyla 2006-2012 yılları arasında bulunan hataların çok büyük bir kısmı bu kategorilerde bulunan hatalardan meydana geldiğinden sadece D, A ve E kategorileri için kök neden analizi yapılmıştır.

D kategorisinde bulunan tüm kritik ve majör toplam 2804 hatanın projenin hangi fazında bulunduğu ve hatanın nereden kaynaklandığı verisi incelendiğinde Şekil 30'daki gibi bir dağılım oluşmaktadır.

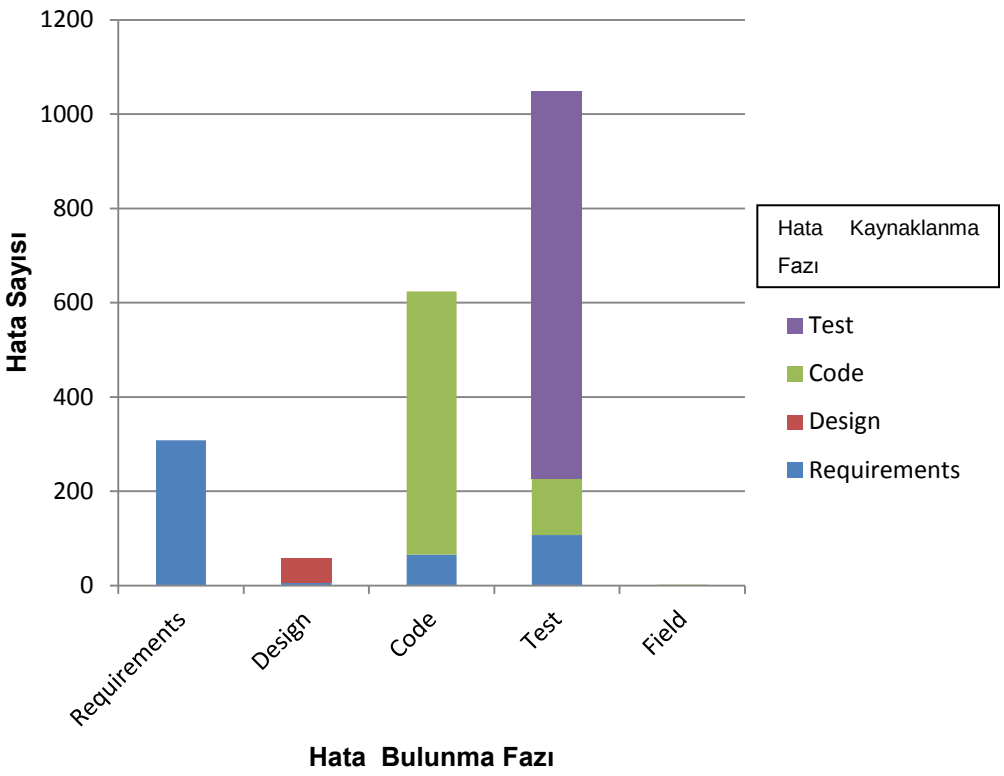


Şekil 30. D Kategorisinde Bulunan Hataların Hangi Fazda Bulunduğu ve Hata Kaynak Noktası (Tüm Projeler)

Şekil 30'da verilen grafikte bir sonraki faza bir önceki fazdan hata aktarımı olmaması istenilen durumdur. Requirements hatalarının bir sonraki faz olan Design, Code ve Test'te bulunması, projenin maliyetini ve bitiş süresini etkileyen bir durum olacaktır. Her faza ait hatanın o fazda bulunması istenmektedir. Şekil 30'da çok belli olmamasına rağmen Field aşamasında bulunan hatalar da Code ve Testten kaynaklanmaktadır.

D kategorisinde bulunan hataların büyük bir bölümü Test ve Requirements aşamalarından kaynaklanmaktadır. Dolayısı ile bu aşamalarda D kategorisinde bulunan hataların yapılmaması için süreçte iyileştirmeler yapılmalıdır.

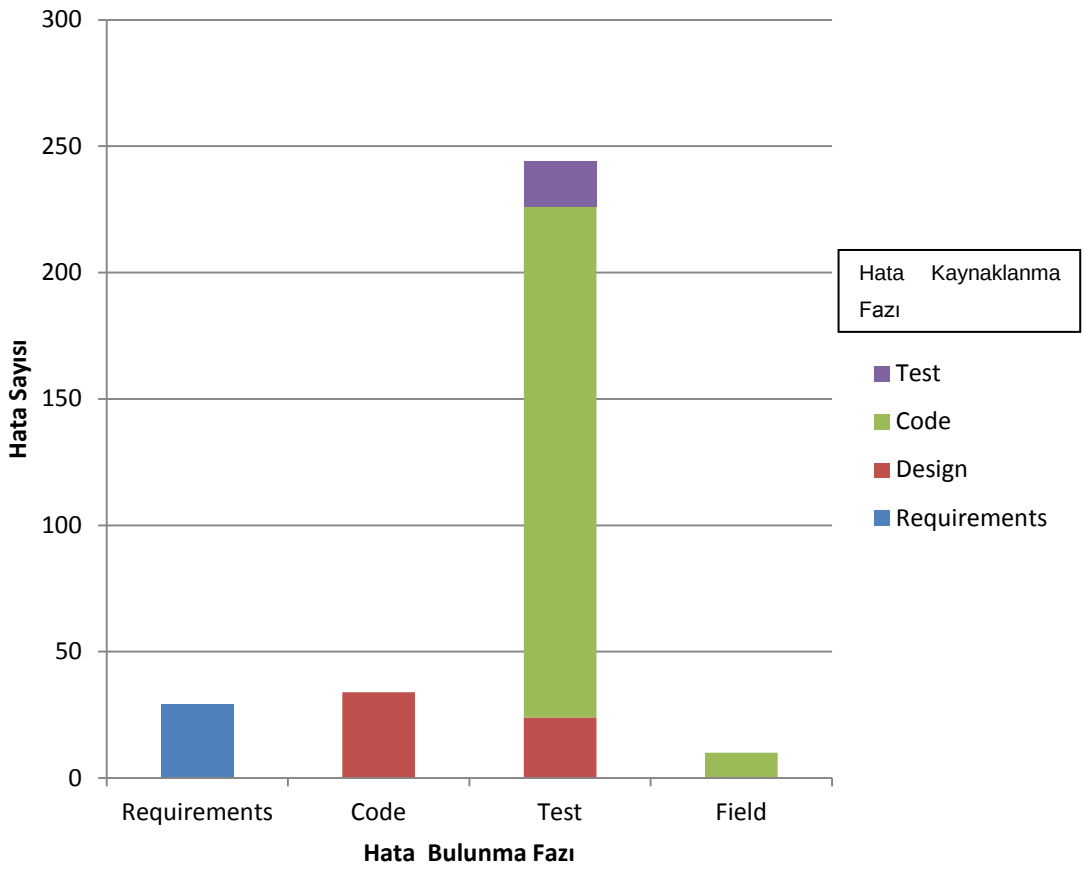
A kategorisinde bulunan tüm kritik ve majör toplam 2160 hatanın, projenin hangi fazında bulunduğu ve hatanın nereden kaynaklandığı verisi incelendiğinde Şekil 31'deki gibi bir dağılım oluşmaktadır.



Şekil 31. A Kategorisinde Bulunan Hataların Hangi Fazda Bulunduğu ve Hata Kaynak Noktası (Tüm Projeler)

Şekil 31'de de görüldüğü gibi bu hata kategorisinde bulunan hatalar en çok Test aşamasında, daha sonra Code aşamasında yapılmaktadır. Test ve Code aşamalarında A kategorisine daha çok dikkat edilmesi gerektiği sonucuna varılabilmektedir.

E kategorisinde bulunan tüm kritik ve majör toplam 647 hatanın projenin hangi fazında bulunduğu ve hatanın nereden kaynaklandığı verisi incelendiğinde Şekil 32'deki gibi bir dağılım oluşmaktadır.



Şekil 32. E Kategorisinde Bulunan Hataların Hangi Fazda Bulunduğu ve Hata Kaynak Noktası (Tüm Projeler)

E kategorisinde yapılan hataların en fazla Code fazında yapıldığı, Design fazında hiç E kategorisine ait hata ile karşılaşılmadığı görülmektedir.

Çizilen Pareto grafikleri yardımıyla tüm hataların en çok 3 kategoride toplandığı görülmektedir. Şirket çalışanları ile ortak alınan bir kararla kök neden analizi ile incelenmesi gereken hata kategorilerinin A,D,E kategorileri olması gerektiği görüşüne varılmıştır. Bu kategorilerde bulunan majör ve kritik hata toplamı 5611 adettir. Tüm projelerde toplam kritik ve majör hata sayısı 7509 olarak görülmektedir. Bu iki değeri oranladığımızda kritik ve majör hataların %74'ünün A, D ve E kategorilerinde bulunan hatalardan kaynaklandığı görülmektedir.

Kök neden analizi aşamasında belirlenen A, E ve D verileri projelerde çalışan elemanlar ile yapılan toplantılar ile görüşülmüştür. Pareto grafikleri aracılığıyla tespit edilen 3 ana problem için, 6 ayrı toplantı yapılmıştır. Toplantılara toplam 24

kişinin katılımı olmuştur ve kök nedenlerin belirlenmesi için 30 saat beyin fırtınası yapılmıştır.

Kök neden analizi yapılırken kurallar;

- Problemin kök nedeni olacak en önemli 10 sebep seçilir.
- Belirlenen bu 10 sebebe 10 üzerinden puanlar verilir. En önemli sebebe 10 puan verilmelidir.

Yapılan bu çalışmanın amacı A, D ve E kategorilerinde bulunan hataların sayılarını azaltmak için hangi konulara önem verilmesi gerektiğinin tespit edilmesidir.

4.2.1.1. A kategorisi hataları için kök neden analizi

Yapılan ilk toplantı grubu, A kategorisinde bulunan hataları değerlendirmek üzere yapılmıştır.

Toplantıdaki amaç, A kategorisinde bulunan hataların yüksek sayıda olması sorununun neden kaynaklandığını ortaya çıkarıp, bu sorunun en etkili sebebinin belirlenmesidir.

Bu toplantıya 7 kişi katılmıştır fakat bu kişilerin kimlikleri, şirketin isimleri vermek istememesi nedeniyle gizlenmektedir. Aşağıda, analize katılan kişilerin şirketteki görevleri verilmektedir.

- 3 Yazılım Mühendisi,
- 1 Takım Lideri (Analiz Lideri),
- 1 Kalite Mühendisi,
- 1 Proje Yöneticisi,
- 1 Süreç Mühendisi (Moderatör).

Süreç mühendisi, toplantıda moderatör olarak görev yapmıştır. Takım lideri 1 kişi ise kök neden analizinin lideri olarak görev yapmıştır.

Toplantıya katılan ekip yaklaşık 6'şar saatlik 2 toplantı yaparak A Kategorisinde bulunan tüm hataları incelemişlerdir. Hatalar teker teker ele alınarak, ana sebepler beyin fırtınası yöntemi ile bulunmaya çalışılmıştır. Toplantı esnasında görüşülen

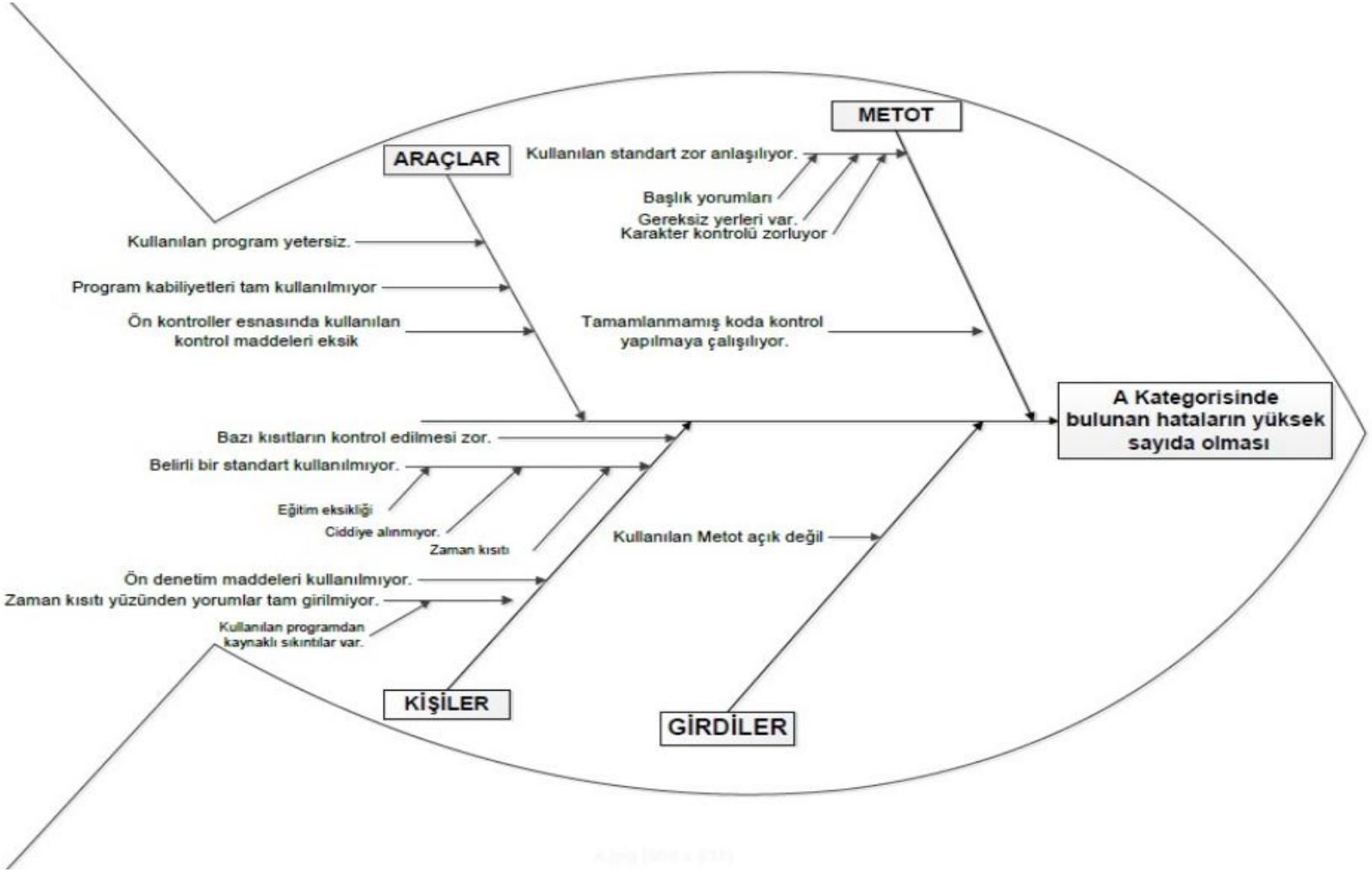
sebepler, sebep-sonuç diyagramına yerleştirilmiştir. Yapılan toplantılar sonucunda ortaya çıkan balık kılçığı diyagramı Şekil 33'de verilmiştir.

Tablo 14. A Kategorisi Kök Neden Numaraları ve Açıklamaları

Kök neden No	A Kategorisinde Bulunan Kök nedenler
1	Bazı kısıtların kontrol edilmesi zor. (K)
2	Belirli bir standart kullanılmıyor. (K)
3	Eğitim eksikliği. (K)
4	Ciddiye alınmıyor. (K)
5	Zaman kısıtı. (K)
6	On denetim maddeleri kullanılmıyor. (K)
7	Zaman kısıtı yüzünden yorumlar tam girilmiyor. (K)
8	Kullanılan programdan kaynaklı sıkıntılar var. (K)
9	Kullanılan standart zor anlaşılıyor. (M)
10	Başlık yorumları. (M)
11	Gereksiz yerleri var. (M)
12	Karakter kontrolü zorluyor (M)
13	Tamamlanmamış koda kontrol yapılmaya çalışılıyor. (M)
14	Kullanılan program yetersiz. (A)
15	Program kabiliyetleri tam kullanılmıyor. (A)
16	On kontroller esnasında kullanılan kontrol maddeleri eksik. (A)
17	Kullanılan Metot açık değil. (G)

Bulunan kök nedenler Tablo 14'te verildiği gibi numaralandırılmıştır. Toplantıya katılan kişiler ilgili numaralandırmaları kullanarak puanlandırmışlardır.

Sebep-sonuç diyagramının çizimi gerçekleştirildikten sonra takımda bulunan kişiler bulunan 17 sebebe tek tek oy vermiştir. Yapılan bu puanlamalar Tablo 15'te bulunmaktadır.



Şekil 33. A Kategorisinde Bulunan Hatalara Ait Sebep-Sonuç Diyagramı

Tablo 15. A Kategorisi Puanlamaları

Kök Neden No	Proje Yöneticisi	Yazılım Mühendisi 1	Yazılım Mühendisi 2	Yazılım Mühendisi 3	Yazılım Mühendisi 4	Kalite Mühendsi	Süreç Mühendisi
1	0	2	2	1	1	1	0
2	6	7	4	1	1	1	0
3	7	3	3	1	5	1	1
4	8	7	4	1	7	1	9
5	4	9	4	1	1	5	10
6	10	6	6	1	2	4	3
7	5	9	3	1	1	1	8
8	0	3	1	1	1	1	6
9	0	8	5	1	2	1	6
10	0	7	7	1	1	1	0
11	0	8	8	1	1	3	0
12	0	4	9	1	4	1	0
13	0	8	10	1	1	1	7
14	2	5	6	1	1	1	5
15	3	3	4	1	1	8	4
16	1	4	5	1	1	1	0
17	9	6	7	1	1	1	2

17 sebebe 10 üzerinden verilen puanlar daha sonra normalize edilmiştir. Yapılan normalizasyon işlemi;

$$\text{Normalize edilmiş veri} = \frac{\text{Kişinin ilgili nedene verdiği not}}{\text{Tüm nedenlere verdiği not}} \times 100 \tag{4.1}$$

normalize edilmiş veri 4.1'de verilmektedir.

A kategorisinde bulunan hataların yüksek sayıda olma problemi için yapılan oylama ve normalizasyon işlemi sonucu Tablo 16'da verilmiştir.

Tablo 16. Normalize Edilmiş A Kategorisi Puanlamaları

Kök Neden No	Proje Yöneticisi	Yazılım Mühendisi 1	Yazılım Mühendisi 2	Yazılım Mühendisi 3	Yazılım Mühendisi 4	Kalite Mühendisi	Süreç Mühendisi	TOPLAM
1	0,00	2,02	2,27	5,88	3,13	3,03	0,00	16,33
2	10,91	7,07	4,55	5,88	3,13	3,03	0,00	34,56
3	12,73	3,03	3,41	5,88	15,63	3,03	1,64	45,34
4	14,55	7,07	4,55	5,88	21,88	3,03	14,75	71,70
5	7,27	9,09	4,55	5,88	3,13	15,15	16,39	61,46
6	18,18	6,06	6,82	5,88	6,25	12,12	4,92	60,23
7	9,09	9,09	3,41	5,88	3,13	3,03	13,11	46,74
8	0,00	3,03	1,14	5,88	3,13	3,03	9,84	26,04
9	0,00	8,08	5,68	5,88	6,25	3,03	9,84	38,76
10	0,00	7,07	7,95	5,88	3,13	3,03	0,00	27,06
11	0,00	8,08	9,09	5,88	3,13	9,09	0,00	35,27
12	0,00	4,04	10,23	5,88	12,50	3,03	0,00	35,68
13	0,00	8,08	11,36	5,88	3,13	3,03	11,48	42,96
14	3,64	5,05	6,82	5,88	3,13	3,03	8,20	35,74
15	5,45	3,03	4,55	5,88	3,13	24,24	6,56	52,84
16	1,82	4,04	5,68	5,88	3,13	3,03	0,00	23,58
17	16,36	6,06	7,95	5,88	3,13	3,03	3,28	45,70

Her madde için verilen toplam puan tespit edilerek, en yüksek puanı toplayan madde, problemin ana sebebi olarak değerlendirilmektedir.

Tablo 16'dan da görüldüğü üzere, 4 numaralı sebep en yüksek puanı toplamıştır. 4 numaralı sebep "Yapılan işin ciddiye alınmaması"dır. Dolayısıyla yapılan kök neden analizi sonucunda A kategorisinde bulunan hata sayılarının yüksek olmasının ana sebebi "Yapılan işin ciddiye alınmaması" olarak saptanmıştır (Şekil 33).

4.2.1.2. E kategorisi hataları için kök neden analizi

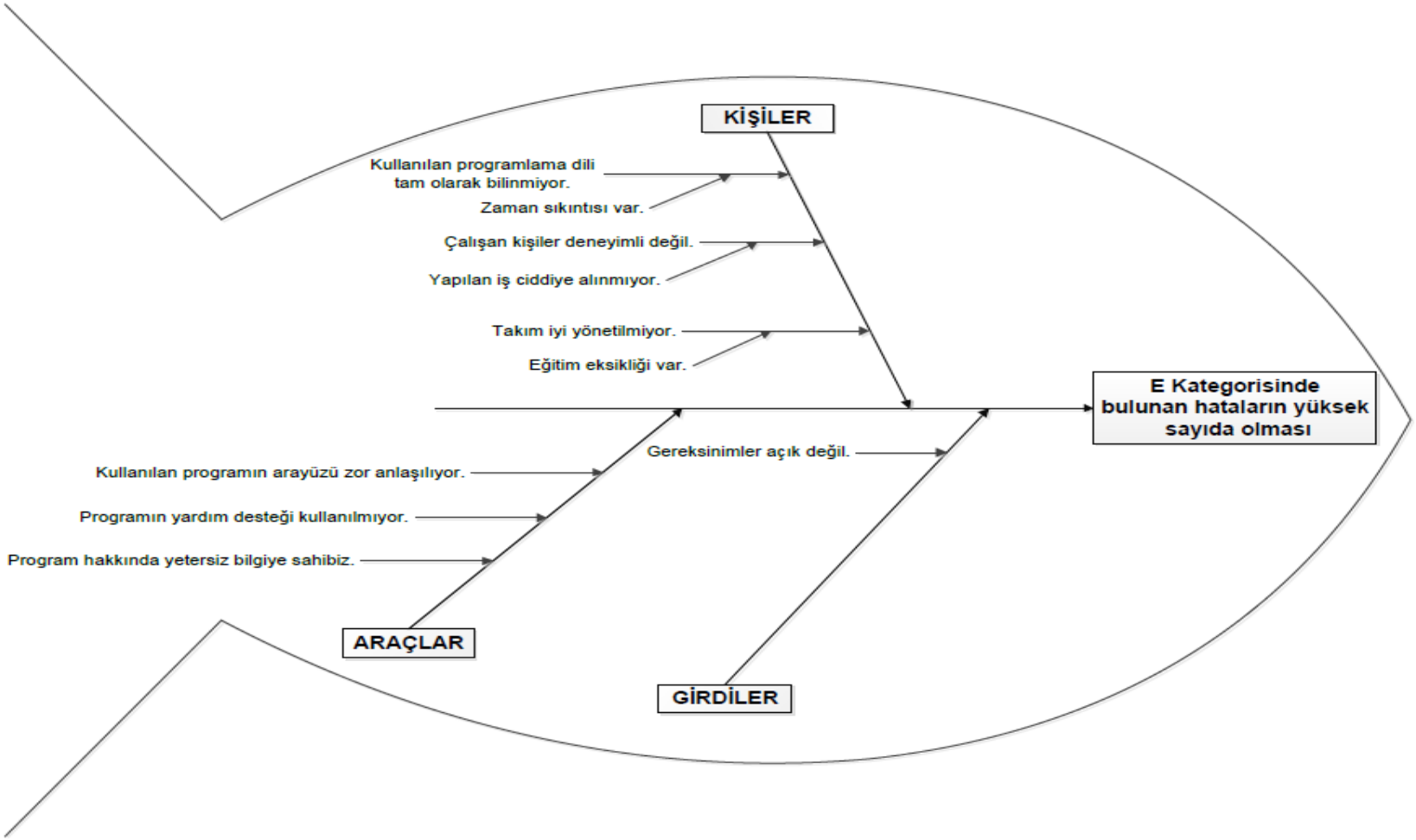
Pareto grafiklerinden elde edilen sonuçlar doğrultusunda seçilen ikinci hata kategorisi E kategorisidir. E kategorisinde bulunan hataları değerlendirmek üzere yapılan toplantılara katılan kişilerin isimleri, şirketin bu isimleri vermek istememesi nedeniyle gizlenmektedir. Fakat analize katılan kişilerin görevleri aşağıda verilmektedir.

E kategorisinde bulunan verilerin kök neden analizini yapmak için toplantılara,

- 1 Yazılım Mühendisi,
- 2 Test Mühendisi,
- 2 Kalite Mühendisi,
- 1 Konfigürasyon Uzmanı (Analizin Lideri)
- 1 Süreç Mühendisi (Moderatör)
- 1 Proje Yöneticisi, katılmıştır.

Süreç mühendisi, toplantıda moderatör olarak görev yapmıştır. Konfigürasyon Uzmanı ise kök neden analizinin lideri olarak görev yapmıştır.

Toplantıya katılan ekip yaklaşık 4'er saatlik 3 toplantı yaparak E Kategorisinde bulunan tüm hataları incelemiştir. Hatalar teker teker ele alınarak ana sebepler beyin fırtınası yöntemi ile bulunmaya çalışılmıştır. Toplantı esnasında görüşülen sebepler balık kılçığı diyagramına yerleştirilmiştir. Toplantılar sonucunda ortaya çıkan sebep-sonuç diyagramı Şekil 34'te verilmektedir.



Şekil 34.E Kategorisinde Bulunan Hatalara Ait Sebep-Sonuç Diyagramı

Tablo 17. E Kategorisi Kök Neden Numaraları ve Açıklamaları

Kök Neden No	A Kategorisinde Bulunan Kök Nedenler
1	Bazı kısıtların kontrol edilmesi zor. (P)
2	Belirli bir standart kullanılmıyor. (P)
3	Eğitim eksikliği. (P)
4	Ciddiye alınmıyor. (P)
5	Zaman kısıtı. (P)
6	On denetim maddeleri kullanılmıyor. (P)
7	Zaman kısıtı yüzünden yorumlar tam girilmiyor. (P)
8	Kullanılan programdan kaynaklı sıkıntılar var. (P)
9	Kullanılan standart zor anlaşılıyor. (M)
10	Başlık yorumları. (M)
11	Gereksiz yerleri var. (M)
12	Karakter kontrolü zorluyor
13	Tamamlanmamış koda kontrol yapılmaya çalışılıyor. (M)
14	Kullanılan program yetersiz. (T)
15	Program kabiliyetleri tam kullanılmıyor. (T)
16	On kontroller esnasında kullanılan kontrol maddeleri eksik. (T)
17	Kullanılan Metot açık değil. (I)

Bulunan kök nedenler Tablo 17'de verildiği gibi numaralandırılmıştır. Toplantıya katılan kişiler, ilgili numaralandırmalar kullanarak puanlandırılmışlardır.

Sebep-sonuç diyagramının çizimi gerçekleştirildikten sonra takımda bulunan kişiler bulunan 10 sebebe tek tek oy vermiştir. Yapılan bu puanlamalar Tablo 18'de bulunmaktadır.

Tablo 18. E Kategorisi Puanlamaları

Kök Neden No	Konfigürasyon Uzmanı	Yazılım Mühendisi 1	Test Mühendisi 1	Test Mühendisi 2	Kalite Mühendisi	Kalite Mühendsi
1	6	8	9	8	9	8
2	4	2	6	3	10	2
3	3	3	5	2	8	3
4	1	4	8	10	7	6
5	7	1	7	7	1	10
6	5	10	10	5	6	9
7	10	7	4	1	3	7
8	9	9	3	6	5	1
9	2	5	2	9	2	4
10	8	6	1	4	4	5

10 sebebe 10 üzerinden verilen puanlar daha sonra normalize edilmiştir. Yapılan normalizasyon işlemi 4.2'de verilen formül ile yapılmıştır.

$$\text{Normalize edilmiş veri} = \frac{\text{Kişinin ilgili nedene verdiği not}}{\text{Tüm nedenlere verdiği not}} \times 100 \quad (4.2)$$

Normalize edilmiş veri Tablo 19'da verilmiştir.

Daha sonra her madde için verilen toplam puan tespit edilerek en yüksek puanı toplayan madde, problemin ana sebebi olarak değerlendirilmektedir. Tablo 19'da normalize edilmiş veriler bulunmaktadır.

Tablo 19. Normalize Edilmiş E Kategorisi Puanlamaları

Kök Neden No	Proje Yöneticisi	Yazılım Mühendisi 1	Yazılım Mühendisi 2	Yazılım Mühendisi 3	Yazılım Mühendisi 4	Kalite Mühendsi	TOPLAM
1	10,91	14,55	16,36	14,55	16,36	14,55	87,27
2	7,27	3,64	10,91	5,45	18,18	3,64	49,09
3	5,45	5,45	9,09	3,64	14,55	5,45	43,64
4	1,82	7,27	14,55	18,18	12,73	10,91	65,45
5	12,73	1,82	12,73	12,73	1,82	18,18	60,00
6	9,09	18,18	18,18	9,09	10,91	16,36	81,82
7	18,18	12,73	7,27	1,82	5,45	12,73	58,18
8	16,36	16,36	5,45	10,91	9,09	1,82	60,00
9	3,64	9,09	3,64	16,36	3,64	7,27	43,64
10	14,55	10,91	1,82	7,27	7,27	9,09	50,91

E kategorisinde bulunan hataların yüksek sayıda olma problemi için yapılan oylama ve normalizasyon işlemi sonucunda Tablo 19'da görüldüğü gibi 1 numaralı sebep en çok puanı toplamıştır. 1 numaralı sebep "Kullanılan programlama dilinin tam olarak bilinmemesi"'dir (Şekil 34). Dolayısıyla yapılan kök neden analizi ile E kategorisinde bulunan hata sayılarının yüksek olmasının ana sebebi, yapılan iş için "Kullanılan programlama dilinin tam olarak bilinmemesi"'dir (Şekil 34). Bu

probleme olası bir çözüm olarak, bu konuda çalışan kişilere, kullandıkları yazılım dilleriyle ilgili eğitim düzenlenmelidir.

4.2.1.3. D kategorisi hataları için kök neden analizi

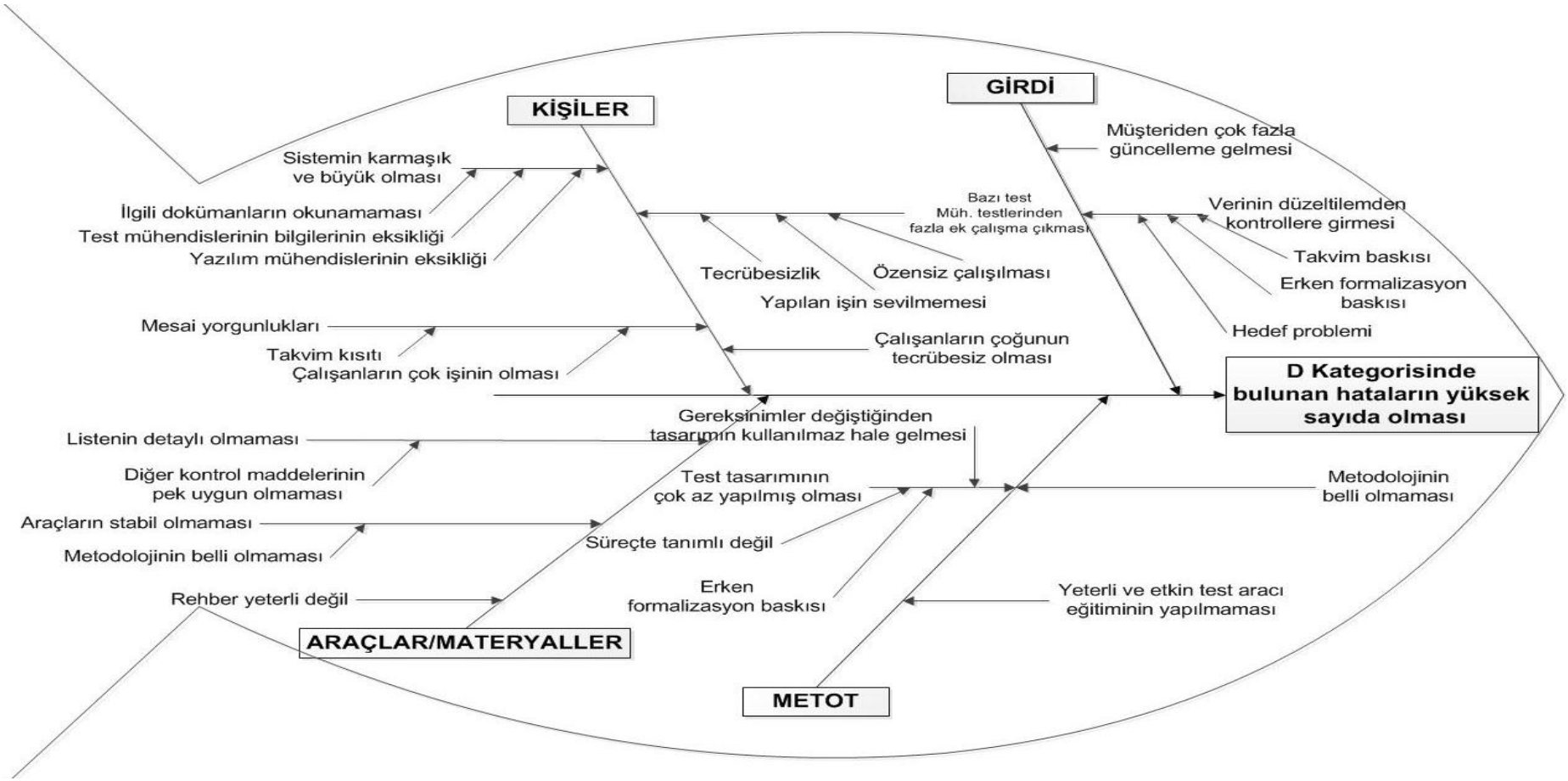
Toplantıdaki amaç, D kategorisinde bulunan hataların yüksek sayıda olması sorununun neden kaynaklandığını ortaya çıkarıp, bu sorunun en etkili sebebinin belirlenmesidir.

Toplantıya katılan kişilerin kimlikleri, şirketin isimleri vermek istememesi nedeniyle gizlenmektedir. Fakat analize katılan kişilerin şirkette bulundukları görevler aşağıda verilmektedir;

- 5 Test Mühendisi,
- 1 Takım Lideri (Analiz Lideri),
- 1 Kalite Mühendisi,
- 1 Proje Yöneticisi,
- 1 Süreç Mühendisi (Moderatör)

Süreç mühendisi, toplantıda moderatör olarak görev yapmıştır. Takım lideri ise kök neden analizinin lideri olarak görev yapmıştır.

Toplantıya katılan ekip yaklaşık 3'er saatlik 2 toplantı yaparak D Kategorisinde bulunan tüm hataları incelemiştir. Hatalar teker teker ele alınarak, ana sebepler beyin fırtınası yöntemi ile bulunmaya çalışılmıştır. Toplantı esnasında görüşülen sebepler sebep-sonuç diyagramına yerleştirilmiştir. Yapılan toplantılar sonucunda ortaya çıkan balık kılçığı diyagramı Şekil 35'te verilmiştir.



Şekil 35. D Kategorisinde Bulunan Hatalara Ait Sebep-Sonuç Diyagramı

Tablo 20. D Kategorisi Kök Neden Numaraları ve Açıklamaları

Kök Neden No	D Kategorisinde Bulunan Kök Nedenler
1	Test mühendislerinin bilgilerinin eksikliği (P)
2	Yazılım mühendislerinin eksikliği (P)
3	Ilgili dokümanların okunamaması (P)
4	Sistemin karmaşık ve büyük olması (P)
5	Çalışanların çok işinin olması (P)
6	Takvim kısıtı (P)
7	Çalışanların çok işinin olması (P)
8	Çalışanların çoğunun tecrübesiz olması (P)
9	Bazı test Müh. Testlerinden fazla ek çalışma çıkması (P)
10	Tecrübesizlik (P)
11	Yapılan işin sevilmemesi (P)
12	Ozensiz çalışılması (P)
13	Listenin detaylı olmaması (M/T)
14	Diğer kontrol maddelerinin pek uygun olmaması (M/T)
15	Araçların stabil olmaması (M/T)
16	Metodolojinin belli olmaması (M/T)
17	Rehber yeterli değil (M/T)
18	Yeterli ve etkin test aracı eğitiminin yapılmaması (M)
19	Metodolojinin belli olmaması (M)
20	Süreçte tanımlı değil (M)
21	Erken formalizasyon baskısı (M)
22	Gereksinimler değiştiğinden tasarımın kullanılmaz hale gelmesi (M)
23	Test tasarımının çok az yapılmış olması (M)
24	Erken formalizasyon baskısı (I)
25	Takvim baskısı (I)
26	Hedef problem (I)
27	Verinin düzeltilemeden kontrollere girmesi (I)
28	Müşteriden çok fazla güncelleme gelmesi (I)

Bulunan kök nedenler Tablo 20'de verildiği gibi numaralandırılmıştır. Toplantıya katılan kişiler ilgili numaralandırmaları kullanılarak puanlandırmışlardır. Materyal ve Araçlar grubunda sayı olarak az neden olmasından dolayı 2 ana kategori birleştirilmiştir.

Sebep-sonuç diyagramının çizimi gerçekleştirildikten sonra, takımdaki kişiler, bulunan 28 nedene tek tek oy vermiştir. Toplantılara katılan 9 kişiden sadece 6 kişi puanlamaya katılmıştır. Yapılan bu puanlamalar Tablo 21'de verilmektedir.

Tablo 21. D Kategorisi Puanlamaları

Kök Neden No	Proje Yöneticisi	Test Mühendisi	Test Mühendisi	Test Mühendisi	Test Mühendisi	Kalite Mühendisi
1	10	8	10	4	9	7
2	3	2	5	5	2	7
3	2	5	8	2	5	6
4	3	8	4	4	6	7
5	6	8	6	3	1	6
6	5	8	6	5	3	10
7	6	8	7	4	0	7
8	7	0	5	2	6	8
9	4	7	0	4	0	4
10	7	7	5	4	0	7
11	6	3	6	4	6	4
12	4	3	1	4	9	4
13	4	8	7	10	10	6
14	3	8	7	10	5	5
15	4	8	6	9	4	7
16	3	8	10	8	6	9
17	5	8	7	7	6	7
18	9	8	6	7	7	10
19	7	10	6	7	8	10
20	7	10	6	7	7	10
21	8	8	4	5	0	5
22	3	10	6	5	3	7
23	1	6	4	10	0	7
24	2	2	7	10	8	5
25	6	6	10	7	1	8
26	0	6	10	5	0	9
27	3	6	10	7	0	9
28	9	9	8	8	0	8

28 sebebe 10 üzerinden verilen puanlar daha sonra normalize edilmiştir. Yapılan normalizasyon işlemi;

$$\text{Normalize edilmiş veri} = \frac{\text{Kişinin ilgili nedene verdiği not}}{\text{Tüm nedenlere verdiği not}} \times 100 \tag{4.3}$$

formülü ile yapılmaktadır. Normalize edilmiş veri Tablo 22'de verilmektedir. Daha sonra her madde için verilen toplam puan tespit edilerek en yüksek puanı toplayan madde, problemin ana sebebi olarak değerlendirilmektedir.

D kategorisinde bulunan hataların yüksek sayıda olma problemi için yapılan oylama ve normalizasyon işlemi sonucu 1 numaralı sebep 31,15 puan ile en çok

oyu toplamıştır. 1 numaralı sebep “Test mühendislerinin bilgisinin eksik olması”’dır (Şekil 35).

Tablo 22. Normalize Edilmiş D Kategorisi Puanları

Kök Neden No	Proje Yöneticisi	Test Mühendisi	Test Mühendisi	Test Mühendisi	Test Mühendisi	Kalite Mühendisi	TOPLAM
1	7,30	4,26	5,65	2,40	8,04	3,52	31,15
2	2,19	1,06	2,82	2,99	1,79	3,52	14,38
3	1,46	2,66	4,52	1,20	4,46	3,02	17,32
4	2,19	4,26	2,26	2,40	5,36	3,52	19,97
5	4,38	4,26	3,39	1,80	0,89	3,02	17,73
6	3,65	4,26	3,39	2,99	2,68	5,03	21,99
7	4,38	4,26	3,95	2,40	0,00	3,52	18,50
8	5,11	0,00	2,82	1,20	5,36	4,02	18,51
9	2,92	3,72	0,00	2,40	0,00	2,01	11,05
10	5,11	3,72	2,82	2,40	0,00	3,52	17,57
11	4,38	1,60	3,39	2,40	5,36	2,01	19,13
12	2,92	1,60	0,56	2,40	8,04	2,01	17,52
13	2,92	4,26	3,95	5,99	8,93	3,02	29,06
14	2,19	4,26	3,95	5,99	4,46	2,51	23,36
15	2,92	4,26	3,39	5,39	3,57	3,52	23,04
16	2,19	4,26	5,65	4,79	5,36	4,52	26,76
17	3,65	4,26	3,95	4,19	5,36	3,52	24,93
18	6,57	4,26	3,39	4,19	6,25	5,03	29,68
19	5,11	5,32	3,39	4,19	7,14	5,03	30,18
20	5,11	5,32	3,39	4,19	6,25	5,03	29,29
21	5,84	4,26	2,26	2,99	0,00	2,51	17,86
22	2,19	5,32	3,39	2,99	2,68	3,52	20,09
23	0,73	3,19	2,26	5,99	0,00	3,52	15,69
24	1,46	1,06	3,95	5,99	7,14	2,51	22,12
25	4,38	3,19	5,65	4,19	0,89	4,02	22,33
26	0,00	3,19	5,65	2,99	0,00	4,52	16,36
27	2,19	3,19	5,65	4,19	0,00	4,52	19,75
28	6,57	4,79	4,52	4,79	0,00	4,02	24,69

Dolayısıyla yapılan kök neden analizi ile D kategorisinde bulunan hata sayılarının yüksek olmasının ana sebebi, “Test mühendisi rolünde çalışan kişilerin, yapılan işi tam olarak bilmemesi” olarak belirlenmiştir. Bu sorunun çözümü olarak, Test Mühendisi rolünde çalışan kişilere eğitim verilmesi gündeme gelmiştir.

4.3. Örnek Olay İncelemesi Sonuçları

Bu çalışmada, belirlenen 3 ana sebep için bulunan kök sebepler değerlendirilmiştir ve bu sebeplerin giderilmesi için nasıl çalışmalar yapılacağı saptanmıştır.

A kategorisinde bulunan sebep olan “Yapılan işin ciddiye alınmaması” sebebinin azaltılması için şirkette çalışan kişilere yapılan iş ve önemi konusunda seminerler düzenlenmesine karar verilmiştir.

E kategorisinde bulunan sebep olan “Kullanılan programlama dilinin tam olarak bilinmemesi” sebebinin azaltılması için personele ilgili dilin eğitimlerinin aldırılması için çalışmalar yapılmaya başlanmıştır.

D kategorisinde bulunan sebep olan “Test mühendislerinin bilgisinin eksik olması” sebebinin azaltılması için, belli konularda uzman mühendislerin ortaya çıkmasının teşvik edilmesi, ekipler arası test bilgisinin paylaşımı için eğitim verilmesi kararı alınmıştır.

5. SONUÇ ve ÖNERİLER

Analiz kapsamında şirket veri tabanında 6 yıl boyunca kayıt altına alınmış, toplam 10.053 adet hata girişi ile çalışma yapılmıştır. Yapılan çalışma sayesinde şirket kapsamında toplanmış olan hataların eksikleri ve yanlışları düzeltilmiştir. Şirketin ölçüm grubu, giriş yapılmış olan tüm veriler üzerinden geçerek verilerin doğruluk, bütünlük ve tutarlılık açısından kontrollerini yapmıştır. Hatalı, tutarsız ve eksik alanlar projelerde çalışan kişiler ile beraber teker teker düzeltilmiştir. Verilerin düzeltilmesinde 7 kişi görev almıştır. Toplam 315 hata verisinde yanlış alan girişi yapıldığı tespit edilerek, düzeltmeler yapılmış ve veriler analize hazır hale getirilmiştir.

Kök neden analizleri kapsamında bölüm 2.5'te anlatıldığı gibi öncelikli olarak verilerin değerlendirilmesi, problemin sebeplerinin belirlenmesi için analiz edilecek olan kategorilerin belirlenmesi gerekmektedir. Bu araştırma kapsamında ilk amaç DHS kullanılarak verilerin sınıflandırılması ve kök neden analizi yapılması olarak belirlenmişti. Daha sonraki aşama olarak, şirketin analizden sonra hata değerlendirmelerinde bu sınıflandırma yöntemlerini kullanması amaçlanmaktaydı.

Toplanmış olan 6 yıllık verinin bu sınıflandırma yöntemine dönüştürülmesi sırasında sorunlar yaşandı. Bunun sebebi, toplanmış olan verinin DHS yöntemini uygulamaya elverişli olmamasıydı. Verilerin giriş alanları arasında, DHS yönteminin temelini oluşturan tetikleyiciler (trigger) bulunmamaktaydı. Dolayısı ile şirkette daha önceden geliştirilmiş olan hata sınıflandırma ve analiz metodunun bu analiz kapsamında kullanılmasına karar verildi. Kullanılan metot sonrasında kök neden analizi yapılması kararlaştırıldı.

Yapılan çalışma sonucunda şirketin alt yapısı ve hatalar hakkında olan karakteristiği ortaya çıkmıştır. Müşteriye ürün teslimatı yapılmadan önce yapılan ön kontroller sayesinde hataların büyük bir bölümü bulunarak, ortadan kaldırılmaktadır. Fakat kontroller sonrasında ortaya çıkan (SCR) ve genellikle majör ve kritik olarak nitelendirildiği hatalar Şekil 20'de görülmektedir. İlgili hatalar için geriye dönük çalışma maliyeti çok yüksek olabildiğinden, şirket açısından dikkat edilmesi gereken bir durum olarak göze çarpmaktadır. Dolayısıyla yapılan

hatalar hangi fazlardan kaynaklanıyorsa, bu konu ile ilgili eğitimler veya uzmanlaşmış eleman alımı sayesinde hatalar azaltılabilir.

Hataların tutulduğu kategori tipleri (şirket isteği üzerine kategori isimleri açık bir biçimde verilememektedir) Pareto grafikleri yardımıyla incelendiğinde yapılan hata sayısının en fazla A, E ve D kategorilerinde olduğu görülmüştür. Belirlenen 3 hata kategorisinin toplam hata sayısının %74'ünü yansıttığı belirlenerek, kök neden analizi bu 3 kategori üzerinde yapılmıştır. Kök neden analizi sırasında yapılan toplantılar ile veriler değerlendirilmiş ve hataların kaynaklarını oluşturan sebepler belirlenmiştir.

A kategorisinde bulunan hataların kök sebebi; yapılan işin ciddiye alınmaması, E kategorisinde bulunan hataların kök sebebi; kullanılan programlama dilinin tam olarak bilinmemesi, D kategorisinde bulunan hataların kök sebebi; test mühendislerinin bilgisinin eksik olması olarak tespit edilmiştir. Bulunan bu 3 kök sebep için alınan önlemler ise aşağıda verilmektedir;

- Şirkette çalışan kişilere yapılan iş ve önemi konusunda seminerler düzenlenmesine karar verilmiştir.
- Şirkette çalışan kişilere projelerde kullandıkları programlama dilinin eğitimlerinin aldırılması için çalışmalar yapılmaya başlanmıştır.
- Belli konularda uzman mühendislerin ortaya çıkmasının teşvik edilmesi, ekipler arası test bilgisinin paylaşımı için eğitim verilmesi kararı alınmıştır.

Yapılan analiz şirket adına verilerin toparlanması ve düzeltilmesi için bir başlangıç olarak sayıldı ve üst yönetim ile yapılan bir toplantı aracılığıyla paylaşıldı ve anlatıldı. Yapılan toplantı sonucunda alınan görüşler olumlu yönde oldu. Yapılan analiz, önerilen iyileştirmelerin sürece uygulanmasından sonra tekrar ölçümler yapılarak iyileşmenin olma durumunun kontrolüyle zenginleştirilebilir. Çalışmanın süre kısıtlaması yüzünden şirkette iyileştirmenin hayata geçirilmesi analize yansıtılamamıştır. Şirket ilgili analizi, bünyesinde bulunan projelerin yaşam döngüleri boyunca belirli periyodlarla uygulama kararı almıştır. Bu sayede hataların, proje süreci boyunca azaltılması için çalışılması ve müşteriye sıfıra yakın hata ile ürünü teslim etme hedefi koyulmuştur.

Analizin hata değerlendirmeleri konusunda eksik kalan noktası, bulunan hataların sınıflandırmaları konusunda belirli bir standarda uyan yöntem kullanılmamasıdır. Şirket kapsamında yapılacak olan çalışmalar sonucunda, hata sınıflandırma metodu farklılaştırılarak DHS metoduna daha yakın bir yöntem uygulanabilir. Çalışmaların devam etmesi, DHS yönteminin ayrıntılı bir şekilde araştırılması ve bu yöntem için alt yapı çalışmalarının başlatılması için karar alınmıştır.

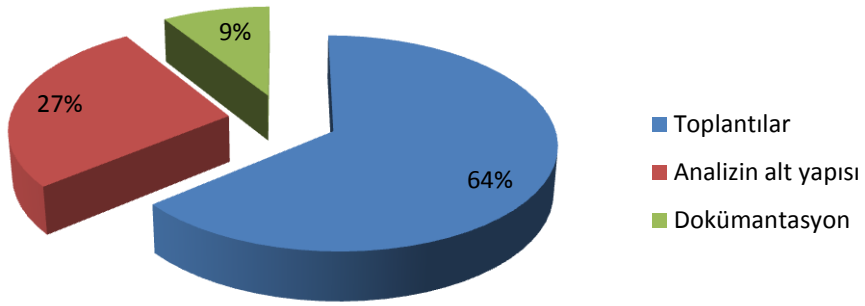
Analiz için eksik bir diğer nokta ise; bulunan hataların sıkıntılı alanlarını gidermek için her projeden 1 kişi ile çalışılmasıdır. Burada kişilerin olaya tam hakim olamadığı noktalar ortaya çıkmıştır. Fakat hataları kayıt altına alan kişiler şirketten ayrıldığı için bulunan verilerin tam doğru yorumlanamamış olma olasılığı bulunmaktadır. Hataların şirket çalışanları tarafından veri tabanına girişi yapılırken daha düzenli ve bilinçli bir şekilde giriş yapılması için eğitimler yapılmaya başlanmıştır.

Bu aşamadan sonra yapılabilecek olan çalışma, DHS yöntemi kullanmak için şirketlerin nasıl bir alt yapısı olması gerektiği, hatalar ile ilgili kurulu bir düzeni ve alt yapısı olan şirketlerin nasıl DHS yöntemi uygulamaya geçirilebileceği konusunda yapılabilir.

Analiz sayesinde, projelerin başarı oranları arttırılabilir. Bunun yanında müşteriye teslim edilen kaliteli ürünler sayesinde, müşteri memuniyetini sağlanmakla kalınmayıp teslim edilen kaliteli ürünler çerçevesinde şirket prestiji arttırılabilir. Yazılım şirketlerinde projelerin belirli standartlara ve düzene uygun yapıldığının belgesi olan CMMI (Capability Maturity Model Integration) düzeyleri arasında rahat geçiş yapmayı kolaylaştırabilir.

5.1. Çalışmanın Efor Analizi

Yapılacak olan diğer çalışmalara ışık tutması açısından analiz kapsamında harcanan efor kayıt altına alınmıştır. Görev alan kişilerin tecrübeleri ve kıdem durumları göz önünde bulundurulmadan harcanan efor toplanmıştır.



Şekil 35.Analiz Kapsamında Harcanan Efor

Analiz süresince şirket çalışanları ile yapılan çalışma yaklaşık olarak 6,5 adam aylık[10] süre zarfında tamamlanmıştır. Analiz kapsamında yapılan toplantılara 24 kişi katılmış, toplantıların tamamı 30 saat sürmüştür. Analizin alt yapısını oluşturmak için harcanan efor 302 saattir. Sunumların hazırlanması ve dokümantasyon işlemi ise 94 saat olarak kayıt altına alınmıştır.

[10] 1 adam ay=180 saat olarak kabul edilmiştir.

KAYNAKÇA

[1] Gouws J. ve Gouws, L.E. Fundementals of Software Engineering Project Management, Malikan Pty Ltd, 9s. 2004.

[2] Ülgen, S. Yazılım Geliştirme Süreçleri, http://www.sulc3.com/surecler.html, 2012.

[3] Freeman, P. Software Perspectives, Kaiforniya: Adison Wesley, 33s.1987.

[4] Sommerville, I. Software Engineering, İngiltere: Pearson Addison Wesley, 5s., 2004.

[5] IEEE Standarts Software Engineering, Volume Four: Resource and Technique

Standartds, Amerika: Institude of Electrical and Electronics Engineers,1999.

[6] Software Engineering Institude, http://www.sei.cmu.edu/solutions/softwaredev/?location=mainnav&source=1358.

16 08 2012.

[7] Brooks, F. P. No Silver Bullet: Essence and Accidents of Software Engineering,

University of North Carolina at Chapel Hill, cilt 20,10-19s, 1987.

[8] Brooks, F. P. No silver Bullet Refired, Software Engineering, no. Anniversary Edition with four new chapters ed. 1995.

[9] McDonald, M. Musson R. ve Smith, R. The Practical Guide to Defect Prevention, Washington: Microsoft Press, 2008.

[10] Boehm, B. W. Industrial Software Metrics: A Top-Ten List, 1987.

http://csse.usc.edu/csse/TECHRPTS/1987/usccse87-503/usccse87-503.pdf.

23.05.2012.

[11] Norris, M. Ve Rigby, P. Sotfware Engineering Explained, John Wiley & Sons Ltd, 1-15s, 1992.

[12] Hayes, F. Chaos is Back, 08.05.2004.

http://www.computerworld.com/s/article/97283/Chaos_Is_Back.,25.05.2012

[13] Rubinstein, D. SD Times, 01.03.2007. http://www.sdtimes.com/link/30247. 10.06.2012.

[14] Briand, L. C. Ve Wieczorek, I. Resource Estimation in Software Engineering,

International Software Engineering Research Network, Technical Report, 1-67s, 2000.

[15] Kulpa M. K. ve Johnson, K. A. Interpretting the CMMI, ABD: Auerbach, 2003.

[16] Eveleens, L.J. ve Verhoef, C, The Rise and Fall of the Chaos Report Figures,

IEEE Software, 30-36s, 2010.

[17] http://www.tdk.gov.tr/index.php?option=com_gts&arama=gts&guid=TDK.GTS.

4ff03c6fb705e5.55935480. 23.05.2012.

[18] http://oxforddictionaries.com/definition/defect?q=defect. 12.05.2012.

[19] Sokullu, E. Tekno İntel, 2012. http://www.teknointel.com/makaleler/ipk.htm.

01.07.2012.

[20] Kumaresh, S. ve Baskaran, R. Defect Analysis and Prevention for Software Process Quality Improvement, International Journal of Computer Aplications, cilt 8, no. 7, 42-47s, 2010.

[21] Rooney J. J. ve Heuvel, V. L. N. Root Cause Analysis For Begginers, Quality Progress, 45-53s, 2004.

[22] Yıldıztekin, İ. Maliyet Kontrolü için Faaliyet Analizi, Atatürk Üniversitesi İktisadi ve İdari Bilimler Dergisi, 181-211s, 2011.

[23] Çolak, T. İstatistiksel Süreç Kontrolü ve Uygulamalar,Yüksek Lisans Tezi, Adana, 23s, 2007.

[24] Uysal, E. Kök Neden analizi ve Kalite Yönetim Sistemindeki Yeri, BUREAU VERITAS, 2004.

[25] Raghu T. ve Vinze, A. A Business Process Context for Knowledge Management, Science Direct, 18s, 31.05.2005.

[26] Özsoy, O. İstatistik, Ankara: Siyasal Kitapevi,13s,2005.

[27] Özcan, S. İstatistiksel Proses Kontrol Tekniklerinden Pareto Analizi ve Çimento Sanayiinde Bir Uygulama, İktisadii ve İdari Bilimler Dergisi, 151-174s, 2011.

[28] Akşit, C. Türk silahlı Kuvvetlerinde Toplam Kalite Uygulamaları-2, Ankara: Genel Kurmay Basımevi, 1999.

[29] Proje Yönetimi Bilgi Birikimi Klavuzu, İstanbul: PMI TR, 210-211s, 2011.

[30] Akın B. ve Öztürk, E. İstatistik Process Kontrol Tekniklerinin Bilgisayar Ortamında Kullanılması, İstanbul, 2005.

[31] Çelikçapa, F. Toplam Kalite Kontrolü ve Bursa Bölgesindeki Kalite Kontrol Uygulamalarına İlişkin Bir Araştırma, Busiad Yayınları, 48-49s, 1993.

[32] Özden, Y. Eğitimde Yeni Değerler, Ankara: Pegema Yayıncılık, 2005.

[33] Bircan H. ve Gedik, H. Tekstil Sektöründe İstatistiksel Proses Kontrol Teknikleri Uygulaması Üzerine Bir Deneme, C.Ü. İktisadii ve İdari Bilimler Dergisi, cilt 2, no. 4, 69-79s, 2003.

[34] Jalote, P. Ve Agrawal, N. Using Defect Analysis Feedback For Improving Quality and Productivity In Iterative Software Development., Information and Communications Technology, Enabling Technologies for the New Knowledge Society: ITI 3rd International Conference, 5-6 Dec. 2005

[35] Söylemez, M. Tarhan, A ve Dikici A, Dikey Hata Sınıflandırması (DHS) ile Yazılım Hatalarının Kök Sebeplerinin İncelenmesi, 6. Ulusal Yazılım Mühendisliği Sempozyumu, Ankara, 2012.

[36] Leszak, M. Perry, D. E. ve Stoll, D. A case Study in Root Cause Defect Analysis, Proceedings of the 22nd international conference on Software engineering, İrlanda, 2000.

[37] Yüceer, R. E. DO-178B Standardı, Yazılım Geliştirme Sürecine Getirdiği Maliyetler ve Faydaları, EMO 40. DÖNEM SEMPOZYUM ve KONGRELER ELEKTRİK MÜHENDİSLERİ ODASI, Ankara, 2007.

[38] en.wikipedia. org/wiki/Waterfall_model., 12.07.2012.

[39] Özbilgin, İ. G. ve Özlü, M.: bilgigüvenligi.gov.tr. 12.05.2012.

[40] Akşit, C. Türk silahlı Kuvvetlerinde Toplam Kalite Uygulamaları-2, Ankara: Genel Kurmay Basımevi, 1999.

[41] Montgomery, D. Introduction to Statistical Quality Control, New York: John Wiley & Sons, 2009.

[42] http://www.research.ibm.com/softeng/ODC/ODC.HTM.14.07.2012.

[43] http://websearcher.blogcu.com/jira-nedir/5103524.,20.07.2012.

[44] Kartal, M. İstatistiksel Kalite Kontrol, Sivas: Şafak Yayınları, 1999.