



**T.C.  
FIRAT ÜNİVERSİTESİ  
TEKNOLOJİ FAKÜLTESİ  
YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**

**Proje Dokümantasyonu**



**Tez Analiz**

**Proje Ekibi**

**Muhammet Emin Saygı**

**Tarih – Versiyon**

**05.01.2021 – 1.0.0.0**

## **ÖNSÖZ**

Bu projeyi gerçekleştirirken yardımlarını esirgemeyen başta hocam Doç. Dr. Fatih ÖZKAYNAK' a, ailem ve arkadaşlarıma yardımlarından ve desteklerinden dolayı teşekkürü borç bilirim.

**Muhammet Emin Saygı**

## **1. GİRİŞ**

- 1.1** Projenin Amacı
- 1.2** Projenin Kapsamı
- 1.3** Tanımlamalar ve Kısaltmalar

## **2. PROJE PLANI**

- 2.1** Giriş
- 2.2** Projenin Plan Kapsamı
- 2.3** Proje Zaman-İş Planı
- 2.4** Proje Ekip Yapısı
- 2.5** Önerilen Sistemin Teknik Tanımları
- 2.6** Kullanılan Özel Geliştirme Araçları ve Ortamları
- 2.7** Proje Standartları, Yöntem ve Metodolojiler
- 2.8** Kalite Sağlama Planı
- 2.9** Konfigürasyon Yönetim Planı
- 2.10** Kaynak Yönetim Planı
- 2.11** Eğitim Planı
- 2.12** Test Planı
- 2.13** Bakım Planı
- 2.14** Projede Kullanılan Yazılım/Donanım Araçlar

## **3. SİSTEM ÇÖZÜMLEME**

- 3.1 Mevcut Sistem İncelemesi**
  - 3.1.1 Örgüt Yapısı
  - 3.1.2 İşlevsel Model
  - 3.1.3 Veri Modeli
  - 3.1.4 Varolan Yazılım/Donanım Kaynakları
  - 3.1.5 Varolan Sistemin Değerlendirilmesi
- 3.2 Gereksenen Sistemin Mantıksal Modeli**
  - 3.2.1 Giriş

- 3.2.2 İşlevsel Model
- 3.2.3 Genel Bakış
- 3.2.4 Bilgi Sistemleri/Nesneler
- 3.2.5 Veri Modeli
- 3.2.6 Veri Sözlüğü
- 3.2.7 İşlevlerin Sıradüzeni
- 3.2.8 Başarım Gerekleri

### **3.3 Arayüz (Modül) Gerekleri**

- 3.3.1 Yazılım Arayüzü
- 3.3.2 Kullanıcı Arayüzü
- 3.3.3 İletişim Arayüzü
- 3.3.4 Yönetim Arayüzü

### **3.4 Belgeleme Gerekleri**

- 3.4.1 Geliştirme Sürecinin Belgelenmesi
- 3.4.2 Eğitim Belgeleri
- 3.4.3 Kullanıcı El Kitapları

## **4. SİSTEM TASARIMI**

### **4.1 Genel Tasarım Bilgileri**

- 4.1.1 Genel Sistem Tanımı
- 4.1.2 Varsayımlar ve Kısıtlamalar
- 4.1.3 Sistem Mimarisi
- 4.1.4 Dış Arabirimler
  - 4.1.4.1 Kullanıcı Arabirimleri
  - 4.1.4.2 Veri Arabirimleri
  - 4.1.4.3 Diğer Sistemlerle Arabirimler
- 4.1.5 Veri Modeli
- 4.1.6 Testler
- 4.1.7 Performans

### **4.2 Veri Tasarımı**

- 4.2.1 Tablo tanımları
- 4.2.2 Tablo- İlişki Şemaları
- 4.2.3 Veri Tanımları
- 4.2.4 Değer Kümesi Tanımları

### **4.3 Süreç Tasarımı**

#### 4.3.1 Genel Tasarım

#### 4.3.2 Modüller

##### 4.3.2.1 XXX Modülü

###### 4.3.2.1.1 İşlev

###### 4.3.2.1.2 Kullanıcı Arabirimi

###### 4.3.2.1.3 Modül Tanımı

###### 4.3.2.1.4 Modül iç Tasarımı

##### 4.3.2.2 YYY Modülü

#### 4.3.3 Kullanıcı Profilleri

#### 4.3.4 Entegrasyon ve Test Gereksinimleri

### **4.4 Ortak Alt Sistemlerin Tasarımı**

#### 4.4.1 Ortak Alt Sistemler

#### 4.4.2 Modüller arası Ortak Veriler

#### 4.4.3 Ortak Veriler İçin Veri Giriş ve Raporlama Modülleri

#### 4.4.4 Güvenlik Altsistemi

#### 4.4.5 Veri Dağıtım Altsistemi

#### 4.4.6 Yedekleme ve Arşivleme İşlemleri

## **5. SİSTEM GERÇEKLEŞTİRİMİ**

### **5.1. Giriş**

### **5.2. Yazılım Geliştirme Ortamları**

#### 5.2.1 Programlama Dilleri

#### 5.2.2 Veri Tabanı Yönetim Sistemleri

##### 5.2.2.1 VTYS Kullanımının Ek Yararları

##### 5.2.2.2 Veri Modelleri

##### 5.2.2.3 Şemalar

##### 5.2.2.4 VTYS Mimarisi

##### 5.2.2.5 Veritabanı Dilleri ve Arabirimleri

##### 5.2.2.6 Veri Tabanı Sistem Ortamı

##### 5.2.2.7 VTYS'nin Sınıflandırılması

##### 5.2.2.8 Hazır Program Kütüphane Dosyaları

##### 5.2.2.9 CASE Araç ve Ortamları

### **5.3. Kodlama Stili**

- 5.3.1 Açıklama Satırları
- 5.3.2 Kod Biçimlemesi
- 5.3.3 Anlamlı İsimlendirme
- 5.3.4 Yapısal Programlama Yapıları

### **5.4. Program Karmaşıklığı**

- 5.4.1 Programın Çizge Biçimine Dönüştürülmesi
- 5.4.2 McCabe Karmaşıklık Ölçütü Hesaplama

### **5.5. Olağan Dışı Durum Çözümleme**

- 5.5.1 Olağandışı Durum Tanımları
- 5.5.2 Farklı Olağandışı Durum Çözümleme Yaklaşımları

### **5.6. Kod Gözden Geçirme**

- 5.6.1 Gözden Geçirme Sürecinin Düzenlenmesi
- 5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular
  - 5.6.2.1 Öbek Arayüzü
  - 5.6.2.2 Giriş Açıklamaları
  - 5.6.2.3 Veri Kullanımı
  - 5.6.2.4 Öbeğin Düzenlenişi
  - 5.6.2.5 Sunuş

## **6. DOĞRULAMA VE GEÇERLEME**

### **6.1. Giriş**

### **6.2. Sınama Kavramları**

### **6.3. Doğrulama ve Geçerleme Yaşam Döngüsü**

### **6.4. Sınama Yöntemleri**

- 6.4.1 Beyaz Kutu Sınaması
- 6.4.2 Temel Yollar Sınaması

### **6.5. Sınama ve Bütünleştirme Stratejileri**

- 6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme
- 6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme

### **6.6. Sınama Planlaması**

### **6.7. Sınama Belirtileri**

### **6.8. Yaşam Döngüsü Boyunca Sınama Etkinlikleri**

## **7. BAKIM**

7.1 Giriş

7.2 Kurulum

7.3 Yerinde Destek Organizasyonu

7.4 Yazılım Bakımı

7.4.1 Tanım

7.4.2 Bakım Süreç Modeli

## **8. SONUÇ**

## **9. KAYNAKLAR**

## 1.GİRİŞ

### 1.1 Projenin Amacı

Üniversitemizdeki; Fen Bilimleri Enstitüsü için yazılan tezleri belirli kurallar ile analiz eder. Bu kurallar; Fen Bilimleri Enstitüsü “Tez ve Seminer Yazma Aracı” kapsamında sınırlandırılmıştır.

### 1.2 Projenin Kapsamı

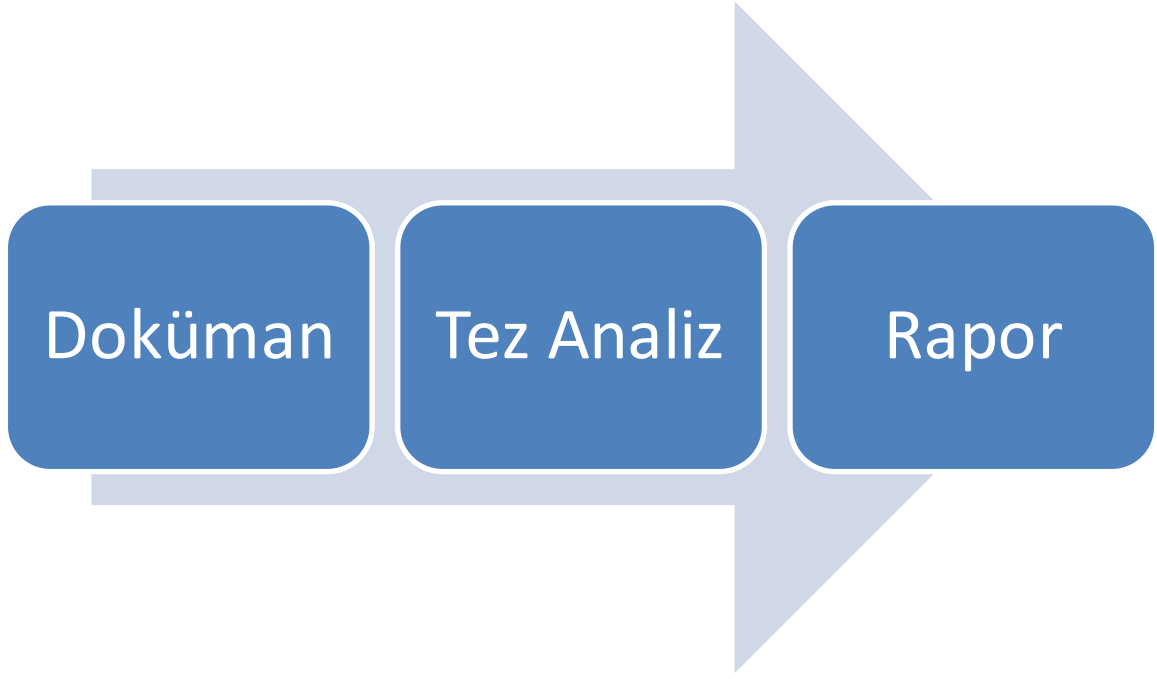
Bu sistem; Tüm çalıştırılabilir işletim sistemlerine uygun oluşturulmuştur.

## 2. PROJE PLANI

### 2.1 Giriş

Analiz yazılımı, belirli kriterler çerçevesinde dokümanları inceler ve bu dokümanlar için gerekli raporları oluşturur.

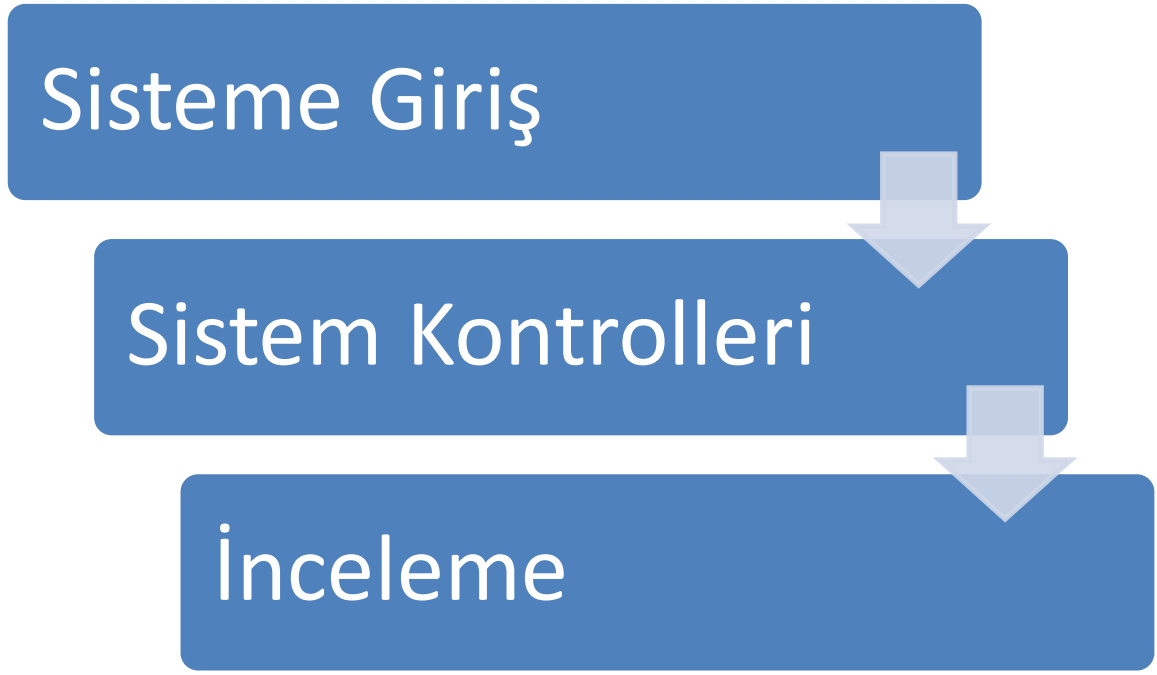




Şekil 2.1 Projenin Genel Yapısı

## 2.2 Projenin Plan Kapsamı

Projenin plan kapsamında genel olarak mevcut sistem, sistemin gerekliliği ve bu sistemin güvenilirliğinden yola çıkıldı. Analiz Yazılımı tamamen pratiklik ve zamandan kazanç odaklı bir oluşumdur. Bu sayede işlemlere büyük ölçüde yardımcı olur.



Şekil 2.2 Projenin Genel İşleyişi

#### Maliyet Kestirim Dokümanı

Ölçüm Parametresi	Sayı	Ağırlık	Toplam
Kullanıcı Girdi Sayısı	5	4	20
Kullanıcı Çıktı Sayısı	5	5	25
Kullanıcı Sorgu Sayısı	4	2	8
Kütük Sayısı	6	4	24
Dışsal Ara yüz Sayısı	7	4	28
Ana işlev Nokta Sayısı			105

Teknik Karmaşıklık Sorusu	Puan
1. Uygulama, güvenilir yedekleme ve kurtarma gerektiriyor mu?	5
2. Veri iletişimi gerekiyor mu?	5
3. Dağıtık işlem işlevleri var mı?	5
4. Performans kritik mi?	4
5. Sistem mevcut ve ağır yükü olan bir işletim ortamında mı çalışacak?	1
6. Sistem, çevrim içi veri girişi gerektiriyor mu?	5
7. Çevrim içi veri girişi, bir ara işlem için birden çok ekran gerektiriyor mu?	1
8. Ana kütükler çevrim-içi olarak mı günleniyor?	5
9. Girdiler, çıktılar, kütükler ya da sorgular karmaşık mı?	3
10. İçsel işlemler karmaşık mı?	2
11. Tasarlanacak kod, yeniden kullanılabilir mi olacak?	4
12. Dönüştürme ve kurulum, tasarımda dikkate alınacak mı?	2
13. Sistem birden çok yerde yerleşik farklı kurumlar için mi geliştiriliyor?	4
14. Tasarlanan uygulama, kolay kullanılabilir ve kullanıcı tarafından kolayca değiştirilebilir mi olacak?	1
TOPLAM	47

1: Çok Az etkisi var

2: Etkisi Var

3: Ortalama Etkisi Var

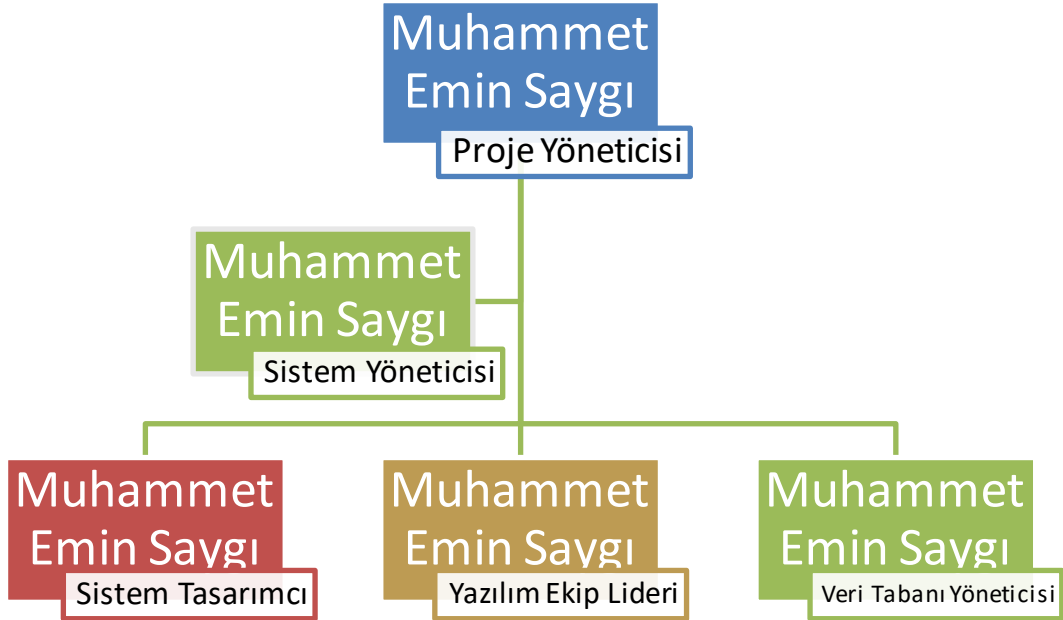
4: Önemli Etkisi Var

5: Mutlaka Olmalı, Kaçınılamaz

## 2.3 Proje Zaman-İş Planı

İş - Zaman Çizelgesi										
Zaman \ İş	1. Hafta	2. Hafta	3. Hafta	4. Hafta	5. Hafta	6. Hafta	7. Hafta	8. Hafta	9. Hafta	10. Hafta
Proje Teklifi	✓									
Proje Planı		●	✓							
Analiz			●	✓						
Sistem Çözümleme				●	●	✓				
Kullanıcı Arayüz Tasarımı						●	✓			
Gerçekleştirim							●	●	✓	
Test									✓	
Sunum										✓

## 2.4 Proje Ekip Yapısı



Proje Yöneticisi	<ul style="list-style-type: none"> <li>•Projenin yönetilmesi</li> <li>•Proje Ekip yapısının oluşturulması</li> <li>•İş planlamasının yapılması</li> </ul>
Sistem Çözümleyici	<ul style="list-style-type: none"> <li>•Dökümantasyon ve Raporlamanın hazırlanması</li> </ul>
Araştırma Ekibi	<ul style="list-style-type: none"> <li>•Proje için gerekli kaynakların temin edilmesi</li> <li>•Örnek sistemlerin incelenmesi</li> </ul>
Veri Tabanı Ekibi	<ul style="list-style-type: none"> <li>•Veri tabanı sistemlerinin oluşturulması</li> <li>•Veri tabanı tasarımının oluşturulması</li> </ul>
Web Tasarımı	<ul style="list-style-type: none"> <li>•Arayüz tanımlamaları çalışmaları</li> <li>•Görsel Tasarım ve Grafik tabanlı yazılım geliştirme araçları ihtiyaçları belirlenmesi</li> <li>•Tasarım ve Kodlama Çalışmaları</li> </ul>
Programcı	<ul style="list-style-type: none"> <li>•Tasarım ve kodlama</li> </ul>
Eğitim Ekibi	<ul style="list-style-type: none"> <li>•Programcı ekibin eğitimi</li> <li>•Kullanıcı eğitimleri</li> <li>•El kitapçıklarının tasarımı</li> </ul>
Test ve Bakım Ekibi	<ul style="list-style-type: none"> <li>•Sistem Testlerinin Yapılması</li> <li>•Sistem bakımının Planın yapılması</li> </ul>

Şekil 2.4.1 Proje Ekip görevleri

## 2.5 Kullanılan Özel Geliştirme Araçları ve Ortamları

Çözümleme Ve Tasarım Araçları	Programlama Araçları	Sınama Araçları
<ul style="list-style-type: none"> <li>• Microsoft Visual 2016</li> <li>• Draw.io</li> </ul>	<ul style="list-style-type: none"> <li>• JAVA</li> <li>• MYSQL</li> </ul>	<ul style="list-style-type: none"> <li>• Windows 7-10</li> <li>• Ubuntu 20.04 LTS</li> <li>• MacOS</li> </ul>

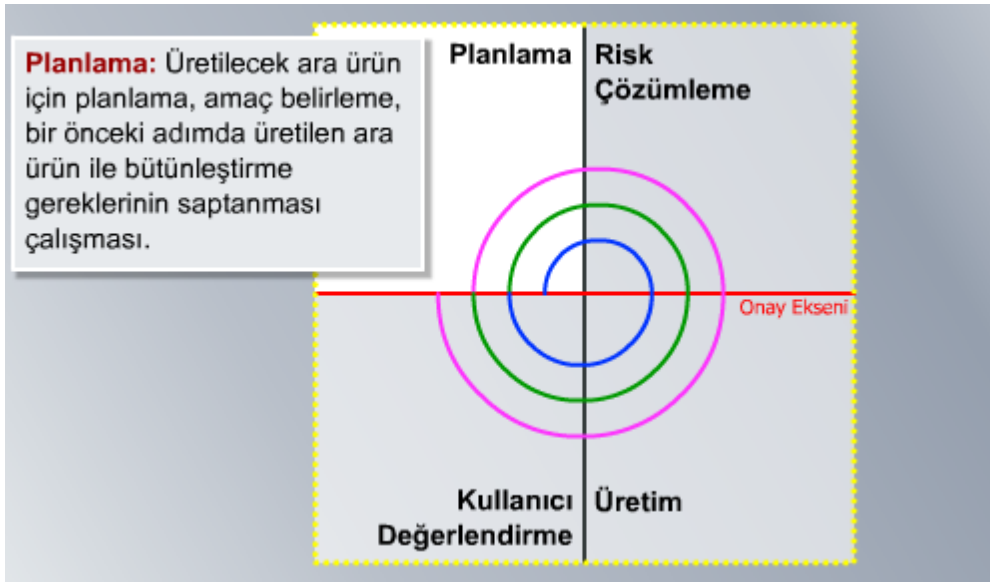
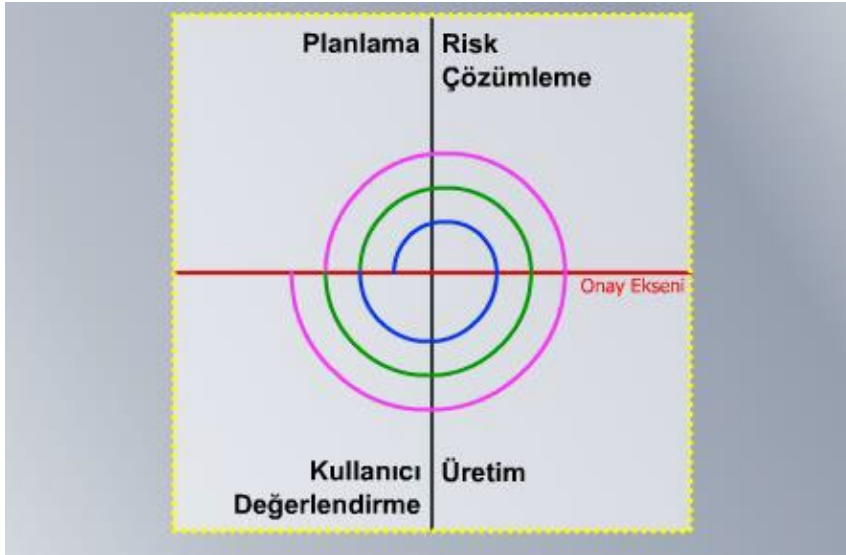
## 2.6 Proje Standartları, Yöntem ve Metodolojiler

Spiralin başladığı ilk çeyrek içinde ilk isterler toplanır ve buna göre proje planlaması yapılır. İkinci çeyrekte, ilk tanımlanan isterlere göre risk çözümlemesi yapılır. Üçüncü çeyrekte, risk çözümlemesi sonunda ortaya çıkan isterlerin tanımlanmasındaki belirsizlikleri ortadan kaldırmak için prototipleşme yöntemi kullanılır. Gerekirse benzetim(benzetim) veya diğer modelleme kullanılarak isterlerin daha sağlıklı tanımlanması sağlanır. Dördüncü çeyrekte, müşteri, ortaya çıkan ilk ürünü inceleyerek değerlendirme yapar, önerilerde bulunur. Bu şekilde tanımlanan ilk döngü bir sonraki döngü için bir girdi oluşturur.

Aşama	Kullanılan Yöntem/Araçlar	Ne İçin Kullanıldığı	Çıktı
<b>Planlama</b>	<ul style="list-style-type: none"> <li>- Veri Akış Şemaları,</li> <li>- Süreç Belirtilimleri,</li> <li>- Görüşme,</li> <li>- Maliyet Kestirim Yöntemleri</li> <li>- Proje Yönetim Araçları</li> </ul>	<ul style="list-style-type: none"> <li>- Süreç İnceleme</li> <li>- Kaynak Kestirimi</li> <li>- Proje Yönetimi</li> </ul>	Proje Planı
<b>Çözümleme</b>	<ul style="list-style-type: none"> <li>- Süreç Belirtilimleri,</li> <li>- Veri Akış Şemaları,</li> <li>- Görüşme,</li> <li>- Nesne İlişki Şemaları,</li> <li>- Veri Sözlüğü</li> </ul>	<ul style="list-style-type: none"> <li>- Süreç Çözümleme</li> <li>- Veri Çözümleme</li> </ul>	Sistem Çözümleme Raporu
<b>Çözümlemeden Tasarıma Geçiş</b>	<ul style="list-style-type: none"> <li>- Akışa Dayalı Çözümleme,</li> <li>- Süreç Belirtilimlerinin Program Tasarım Diline Dönüştürülmesi</li> <li>- Nesne İlişki Şemalarının Veri Tablolarına Dönüştürülmesi</li> </ul>	<ul style="list-style-type: none"> <li>- Başlangıç Tasarım</li> <li>- Ayrıntılı Tasarım</li> <li>- Başlangıç Veri Tasarımı</li> </ul>	Başlangıç Tasarım Raporu
<b>Tasarım</b>	<ul style="list-style-type: none"> <li>- Yapısal Şemalar</li> <li>- Program Tasarım Dili</li> <li>- Veritabanı Tabloları</li> <li>- Veri Sözlüğü</li> </ul>	<ul style="list-style-type: none"> <li>- Genel Tasarım</li> <li>- Ayrıntılı Tasarım</li> <li>- Veri Tasarımı</li> </ul>	Sistem Tasarım Raporu

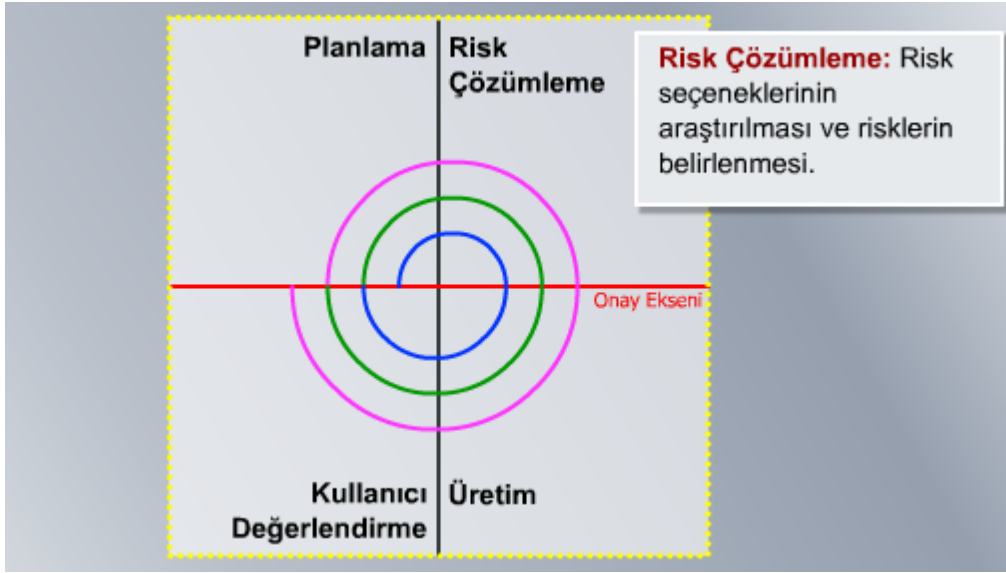
Şekil 2.6 Proje Aşamaları

Proje standartları yukarıda belirtildiği gibidir. Bunu yanı sıra kullanılan sistem modelinde ise helezoni model kullanılmıştır

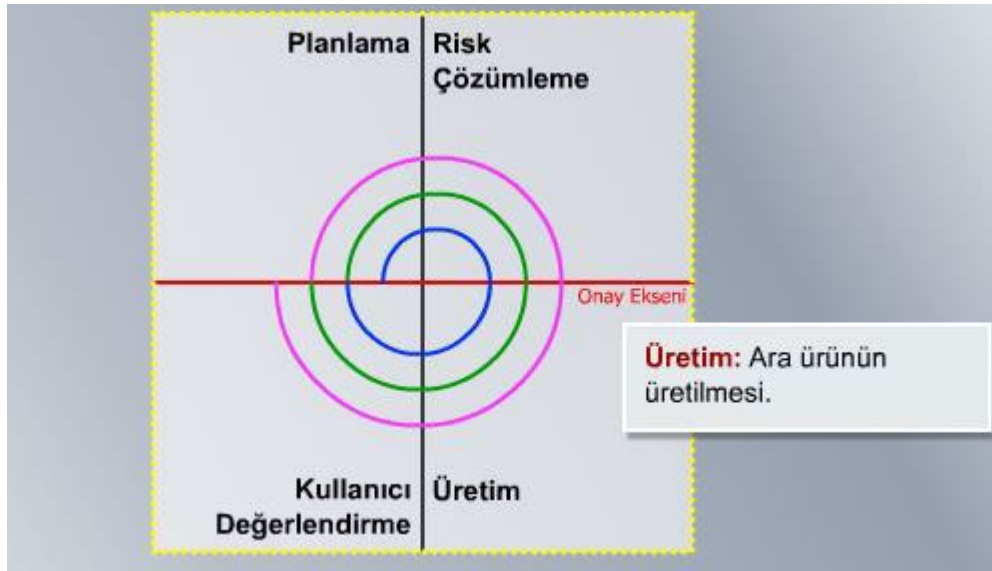


Şekil 2.6.1 Planlama Aşaması

Bu aşamalar yazılım geliştirmede önemli unsurlardır. Bu kullandığımız yazılım geliştirme metodolojilerinden Spiral Modeldir. Bunu kullanmamızın amacı döngü şeklinde kontrol edip hataları en aza indirebilmektir.



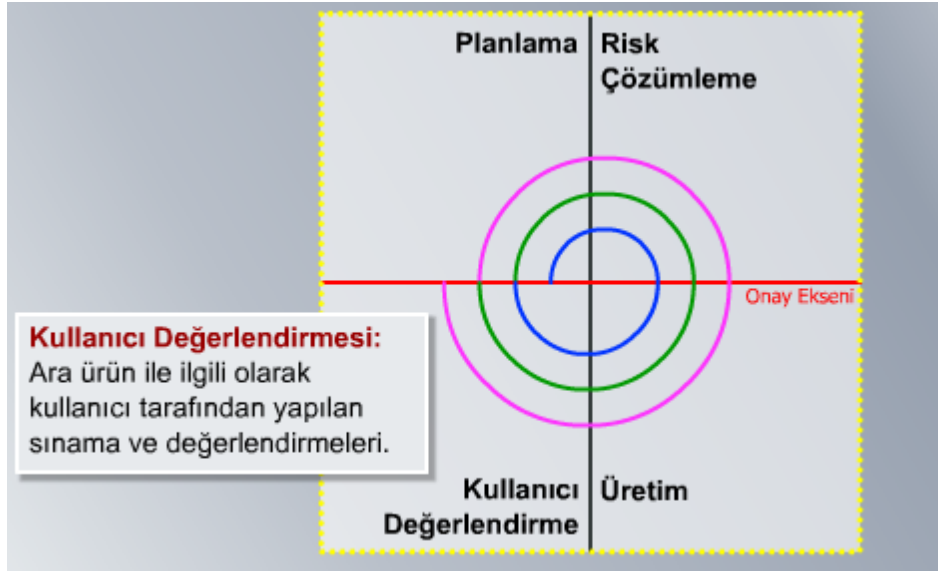
Şekil 2.6.2 Proje Risk Çözümleme Aşaması



Şekil 2.6.3 Proje Üretim Aşaması

Üretim aşaması uzun zaman alan bir aşamadır. Tasarıma yer verilir ve projenin büyük çoğunluğu bitmiş olur. Bu aşamada ara ürün üretilir. Proje bu aşamayla epeyce bir yol almış olacaktır.





Şekil 2.6.4 Kullanıcı Değerlendirme Aşaması

EKİP YAPISI ZAMAN-İŞ PLANI						
AYLAR EKİP	MART		NİSAN		MAYIS	
SİSTEM ÇÖZÜMLEYİCİ	✓	✓				
SİSTEM YÖNETİCİSİ	✓	✓	✓	✓	✓	✓
SİSTEM TASARIMCISI			✓	✓	✓	✓

Şekil 2.6.5 Ekip Yapısı Zaman Planı

## 2.7 Kalite Sağlama Planı



Şekil 2.7 Kalite Sağlama Planı

Projedeki kalite sağlama planımız yukardaki tabloda da belli olduğu üzere;

1.**Ekonomi:** Ekonomik açıdan yazılımın maliyeti her ne kadar ilk seferde pahalı olsa da ileriye dönük düşünüldüğünde ve zaman tasarrufundan ötürü gayet uygundur.

2.**Tamlık:** Projede herhangi bir açık olmamalı ve programda bulunan tüm butonlar textler vs. çalışır ve tamdır.

3.**Yeniden Kullanılabilirlik:** Otomasyon her koşulda tekrardan düzenlenip kullanılabilir.

4.**Etkinlik:** Kullanıcı sistemin her alanına hakim olduğu için sistemi etkin bir biçimde kullanacak.

5.**Bütünlük:** Admin1 sistemin tüm kısımlarına hakim olacak ve program bir bütün halinde çalışacaktır.

- 6.**Güvenilirlik:** Otomasyon gerekli güvenlik önlemlerinin alınması yanı sıra şuan devlet bünyesinde bulunan çok yüksek güvenlik önemli serverlarda saklanacaktır.
- 7.**Modülerlik:** Modülerlik otomasyonun her seviyesindeki kişinin ayrı ayrı sayfalarından söz sahibi olmasını sağlar. Örneğin: Yönetim modülü, Giriş Modülü...
- 8.**Belgeleme:** Bu belgeden de anlaşılacağı üzere tam anlamıyla sistemin özeti olacak bu doküman oluşturulmuştur.
- 9.**Kullanılabilirlik:** Kullanılabilirlik olarak her seviyedeki insana hitap edeceğinden zor renkler karmaşık sistemlerden kaçınılmıştır.
- 10.Temizlik:
- 11.**Değiştirilebilirlik:** Otomasyonun veri tabanını erişme yetkisi olan ve sistem hakkında bilgisi olan herkes sistemde değişiklik yapabilecek.
- 12.**Esneklik:** Proje farklı platformlarda ve internet üzerinden çalışacağından gayet esnektir.
- 13.**Genellik:** Proje her üniversitede kullanılabileceğinden geneldir. Ve Türkiye genelinde kullanılacaktır.
- 14.**Sınanabilirlik:** Projedeki pilot bölge uygulaması sınana bilirliliğinin göstergesidir.
- 15.**Taşınabilirlik:** Sistem internet üzerinden kullanılacağından herhangi bir özel cihaz gerektirmez ve istenilen cihazlarda taşınabilir ve kullanılabilir.
- 16.**Birlikte Çalışılabilirlik:** Bu projedeki en büyük sıkıntı olacak veri girişi şuan var olan ve her bireyin bilgilerinin saklayan sistemle birleşik ve eş zamanlı çalışmakta.

## 2.8 Kaynak Yönetim Planı

Mevcut bir kaynağımız olmadığından kaynak olarak sadece bu proje dokümantasyonu var.

**Kaynak yönetiminde şu hususlar dikkate alınacaktır;**

- Yeterli hata bulunuyor mu?
- Kalite beklendiği gibi mi?
- Eğer yeterli sayıda hata bulunmuyorsa veya Kalite beklenenden daha iyi görünüyorsa, Kalite gerçekten daha iyi olabilir
- Bu projede yeni bir süreç iyileştirmesi kullanılıyor mu?
- Proje elemanları yeni bir eğitim aldı mı?
- Yeni bir araç kullanılıyor mu?
- Yeterli vakit harcanıyor mu?
- İnceleyiciler yeterli hazırlık yapıyor mu?
- İnceleme toplantısında çözüm bulmak için zaman kaybediliyor mu?

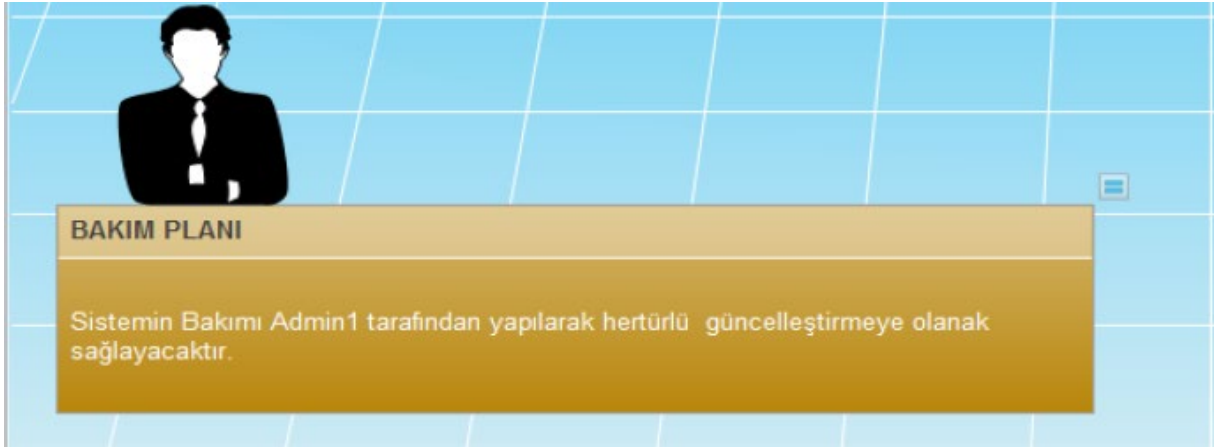
## 2.9 Test Planı

Proje test ekipleri ve görevleri şu şekildedir;

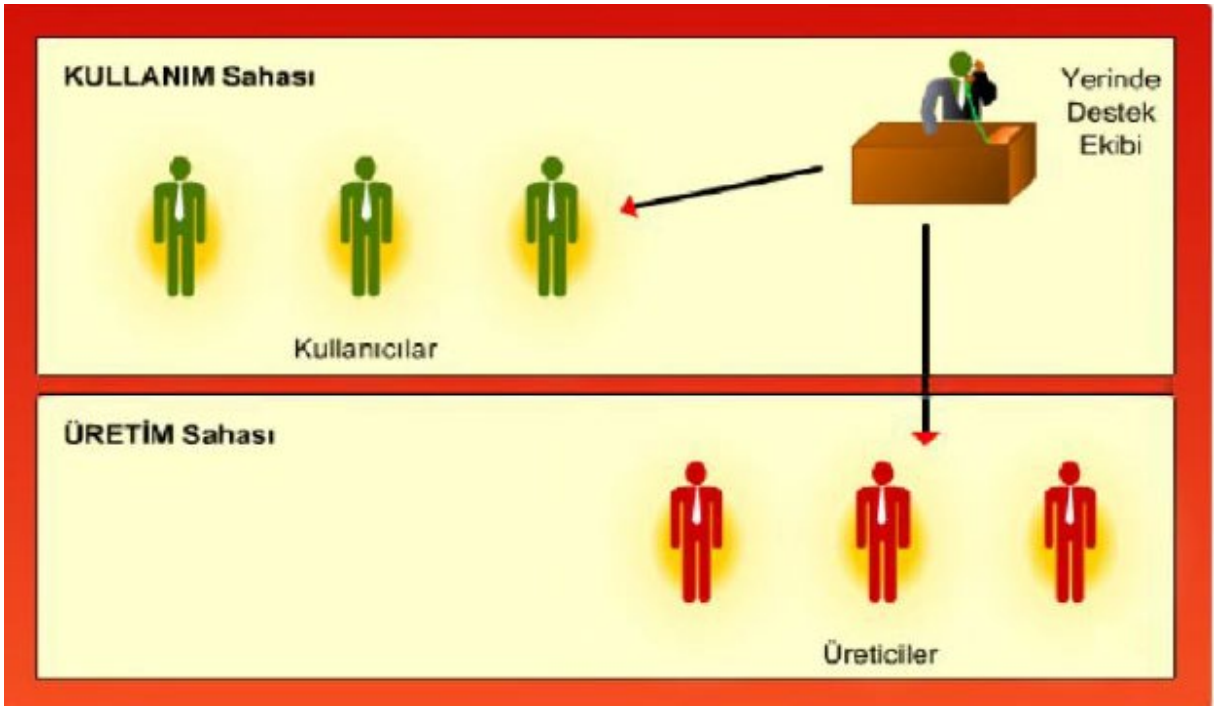
- Ev kullanıcıları görev yapacak.
- Bizzat ben de bu ekibin başında olacağım.

## 2.10 Bakım Planı

Projenin bakım planına gelecek her gün kullanılacak bu sistem tüm değişim ve bazı durumlarda kullanıcı eklenip çıkarılacak tüm bu sistemsel değişiklikler bakım planında yapılacaktır.



Şekil 2.10 Proje Bakım Planı



Şekil 2.10.1 Üretim Sahası

### 3. SİSTEM ÇÖZÜMLEME

### 3.1 Mevcut Sistem İncelemesi

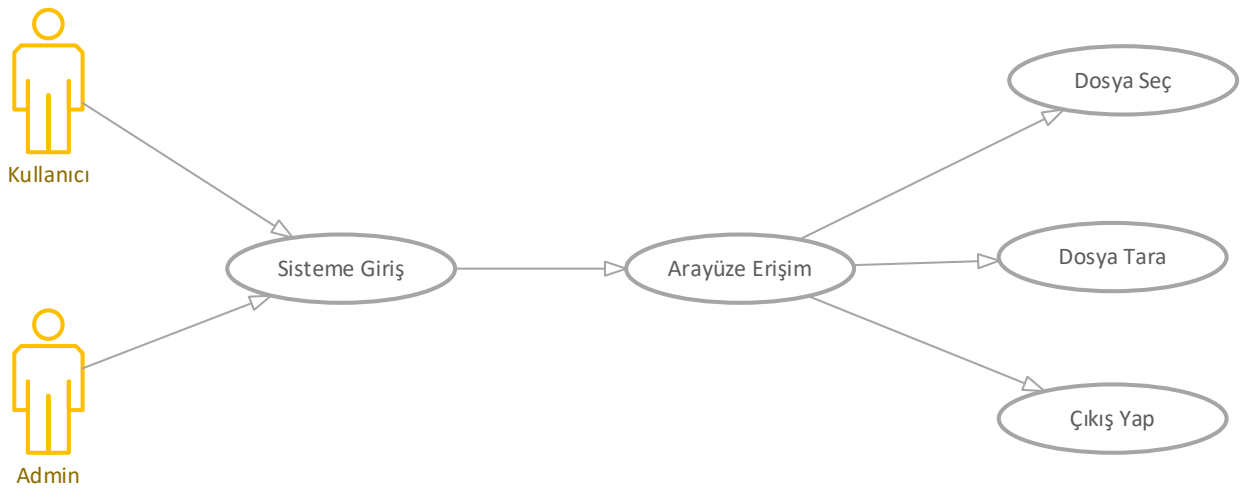
Mevcut sistem incelemesi bireysel olarak yapılacağı için kolay olacaktır.

### 3.2 Gereksenen Sistemin Mantıksal Modeli

#### 3.2.1 Giriş

Mevcut sistemler incelendiğinde sonuca giden yolda epeyce bir eksikler ve resmi olmayan durumlar söz konusu

#### 3.2.2 İşlevsel Model

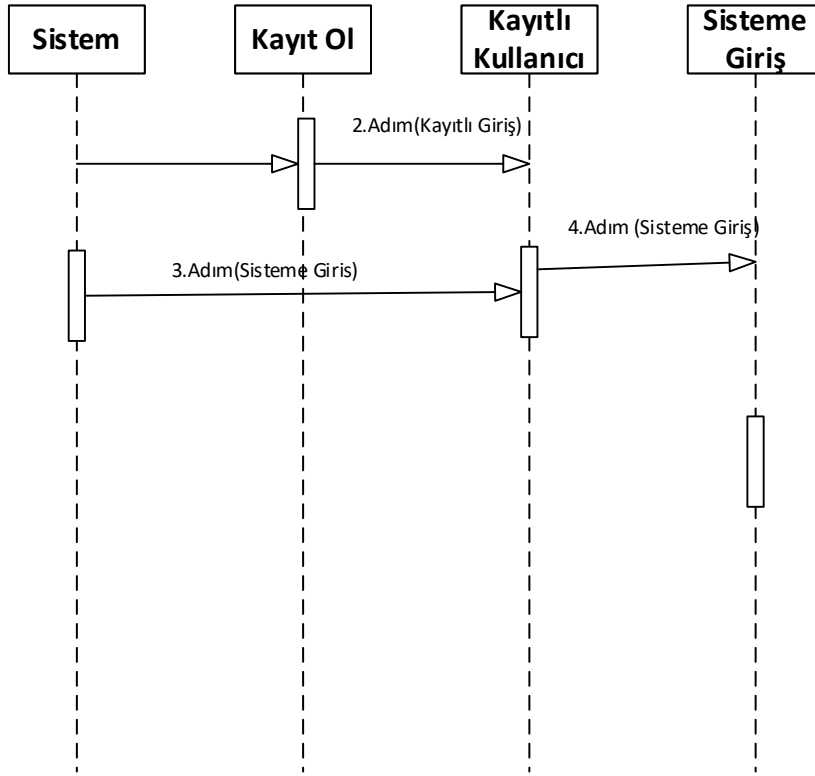


#### 3.2.3 Veri Modeli

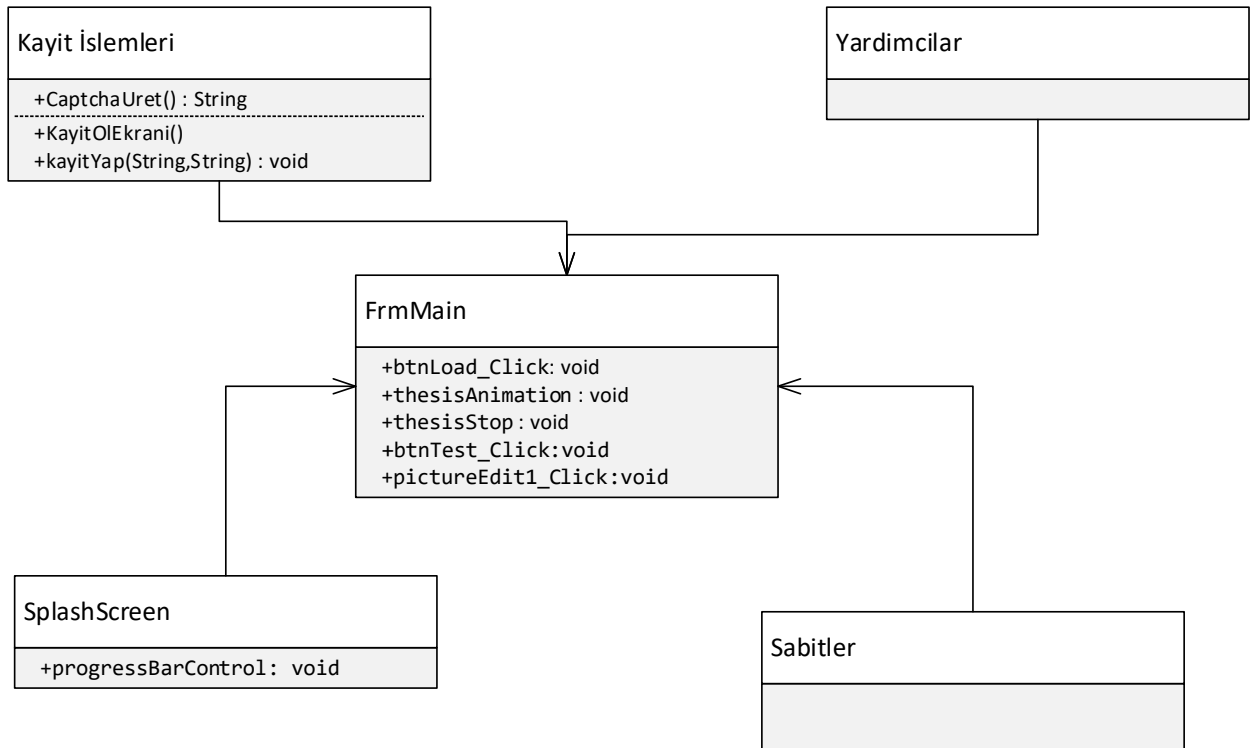
Veri tabanlı ilişkisel veri modelinde veriler tablolar üzerinden kurulan ilişkiye dayanmaktadır

#### 3.2.4 Veri Sözlüğü

#### 3.2.5 İşlevlerin Sıradüzeni



Şekil 3.6 Etkinlik diyagramı



### Şekil 3.7 Sınıf Diyagramı

#### 3.2.6 Başarım Gerekleri

Mevcut sistemler incelendi ve mevcut sistemin eksiklerinden yola çıkılarak, sistemin başarımı için

- Sistemin sonuç üretim doğrulukları
- Tepki sürelerinin en aza indirilmesi
- Mali külfetin azaltılması
- Kullanım kolaylığı
- Anlaşılabilirlik

Temel gereklilikler olarak tespit edilmiştir

#### 3.3 Ara yüz (Modül) Gerekleri

##### 3.3.1 Yazılım Ara yüzü

Projenin çalışması esnasında böyle bir açık verilmemesine özen gösterildi. Gerekli olan her türlü değişiklik source kodları üzerinden yapıp tekrar derlenecek.

##### 3.3.2 Kullanıcı Ara yüzü



# Firat University Thesis Analysis Tool



YÜKLE



TEST ET

# Firat University Thesis Analysis Tool

## %100



C:\Users\Emin\Desktop\Proje Dokümanı\_PD.docx



YÜKLE



TEST ET

-----  
Sayfa Sol kenar boşluğu 3.25 cm olmalıdır!

-----  
Sayfa Üst kenar boşluğu 3.0 cm olmalıdır!

-----  
Declaration bölümü eksik!


-----  
ÖNSÖZ bölümü eksik!

-----  
İÇİNDEKİLER bölümü eksik!


-----  
ÖZET bölümü eksik!


-----  
ABSTRACT bölümü eksik!

-----  
ŞEKİLLER LİSTESİ bölümü eksik!  
-----




**Giriş Yap**

 Kullanıcı Adı

 Şifre

☒ Beni Hatırla



[Şifremi Unuttum](#)

### 3.3.3 İletişim Ara yüzü



### 3.3.4 Yönetim Ara yüzü

Projenin %100 u yönetimsel ara yüzlerden oluşacak. Yöneticinin ekrana girdiğinde karşılaşıacağı ara yüzdür

### 3.4 Belgeleme Gerekleri

#### 3.4.1 Geliştirme Sürecinin Belgelenmesi

Geliştirme sürecinde genel olarak belgelendirilmesi hem ileriye dönük hem de şimdiki geliştirme sürecinde projenin tamamlanma yüzdesini nerede kalınıp nerelerde eksikler olduğunu genel hatlarıyla göstermesi amacıyla yapıldı. Bunun yanı sıra projeye yeni dahil olan personellerin olaya hâkimiyeti açısından bu yönetime başvuruldu.

#### 3.4.2 Eğitim Belgeleri

Mevcut bir belgemiz bulunmamaktadır.

### 3.4.3 Kullanıcı El Kitapları

Bu kısma projenin en son safhasında kullanıcılara verilecek eğitimlerden pilot uygulamalardan yola çıkılarak hazırlanacak. Yani proje sonunda rahat ve kolay kullanımdan dolayı bir eğitim semineri ve bir kullanım kitapçığı hazırlanacaktır.

## 4. SİSTEM TASARIMI

### 4.1 Genel Tasarım Bilgileri

#### 4.1.1 Genel Sistem Tanımı



Şekil 4.1.1 Genel Sistem Tanımı

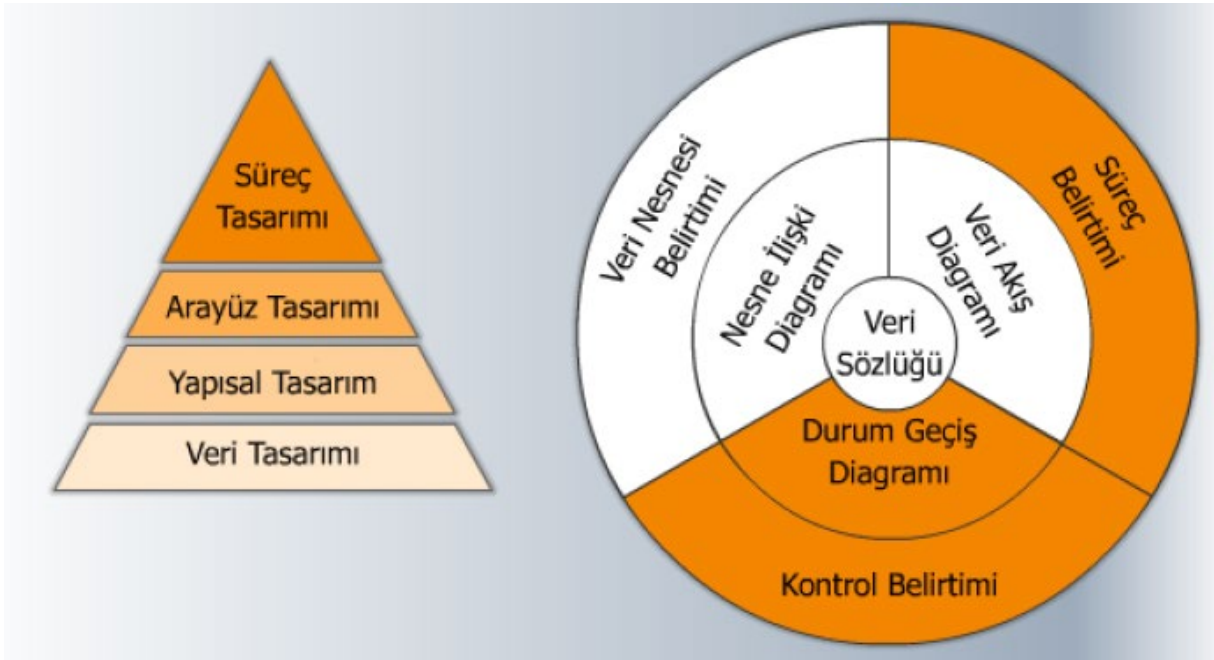
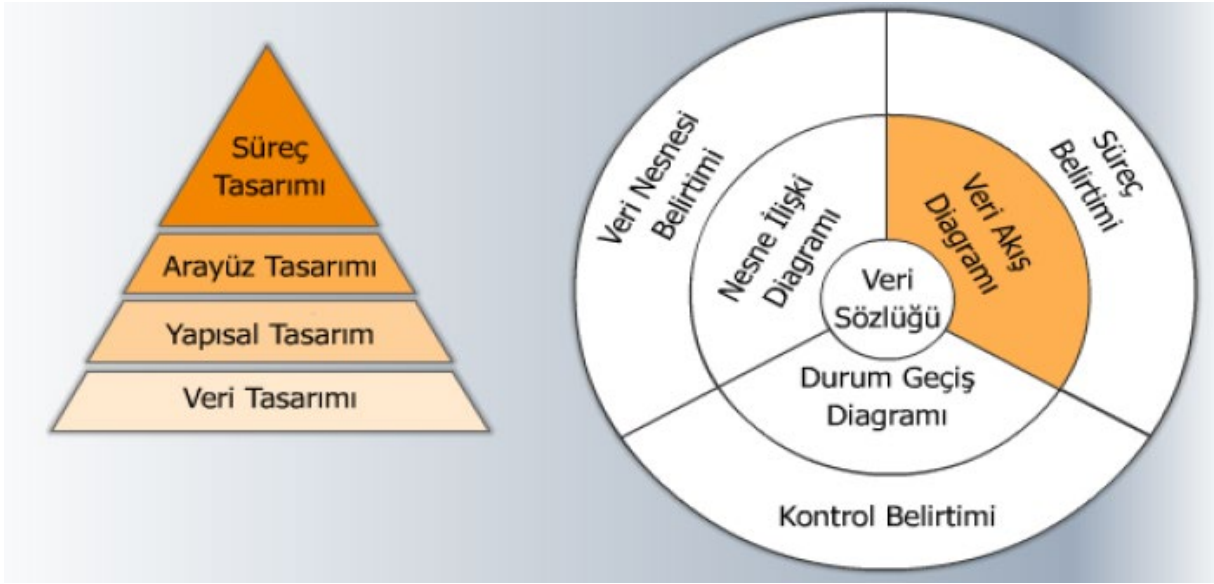
- **Gereksinimler**

Gereksinimler kısmı için anket düzenleyeceğim. Anket sonuçlarına göre hareket edeceğim

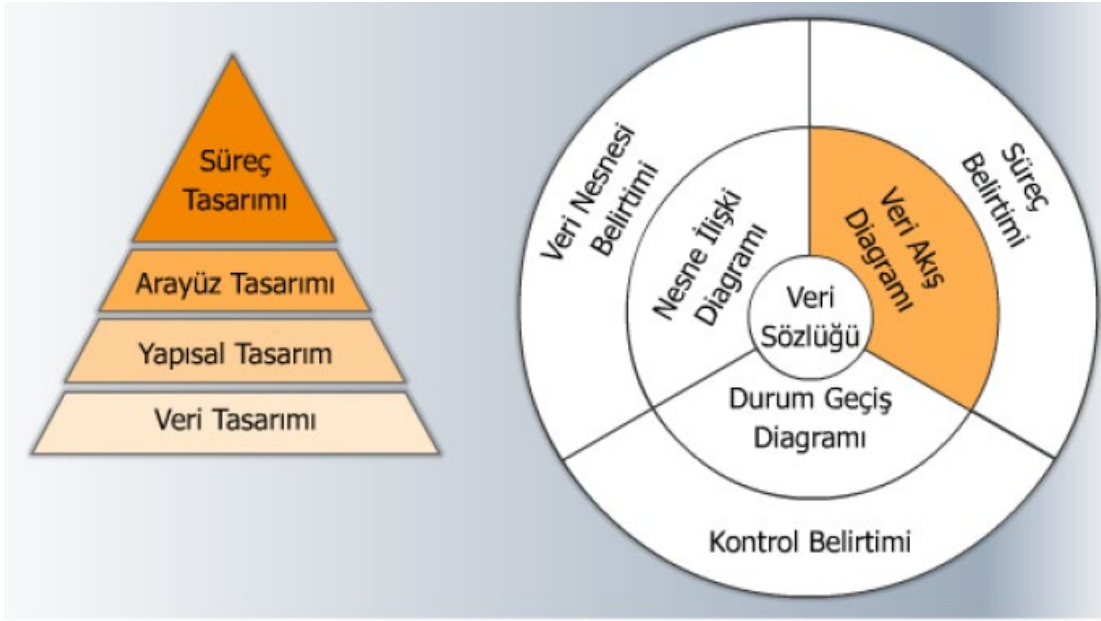
- **Tasarım**

Tasarım aşamasında neler olacağını grafiksel olarak aktarmak isterim.

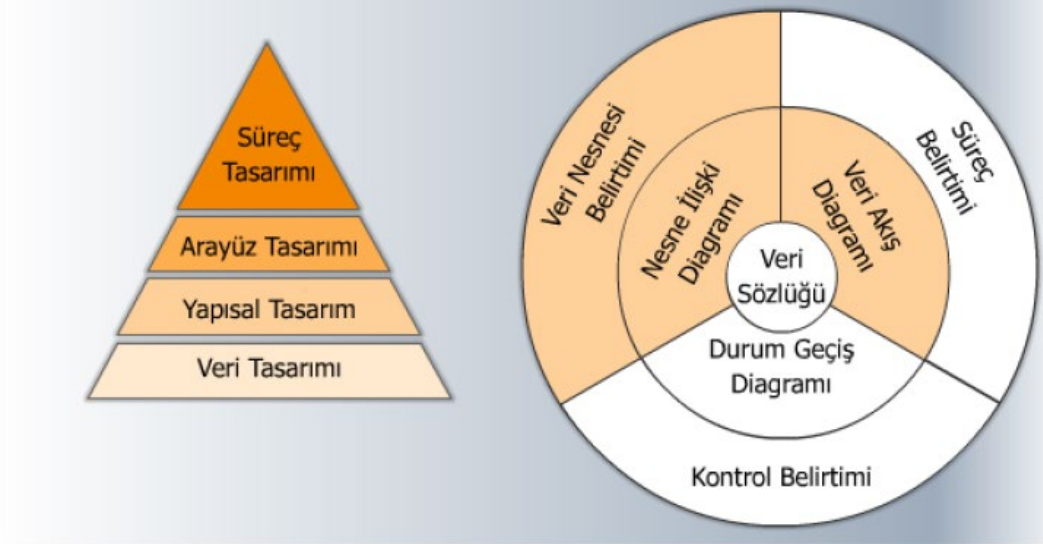
1. İlk önce bir Süreç tasarımı olacak ve adımlar aşağıdaki gibi olacak.



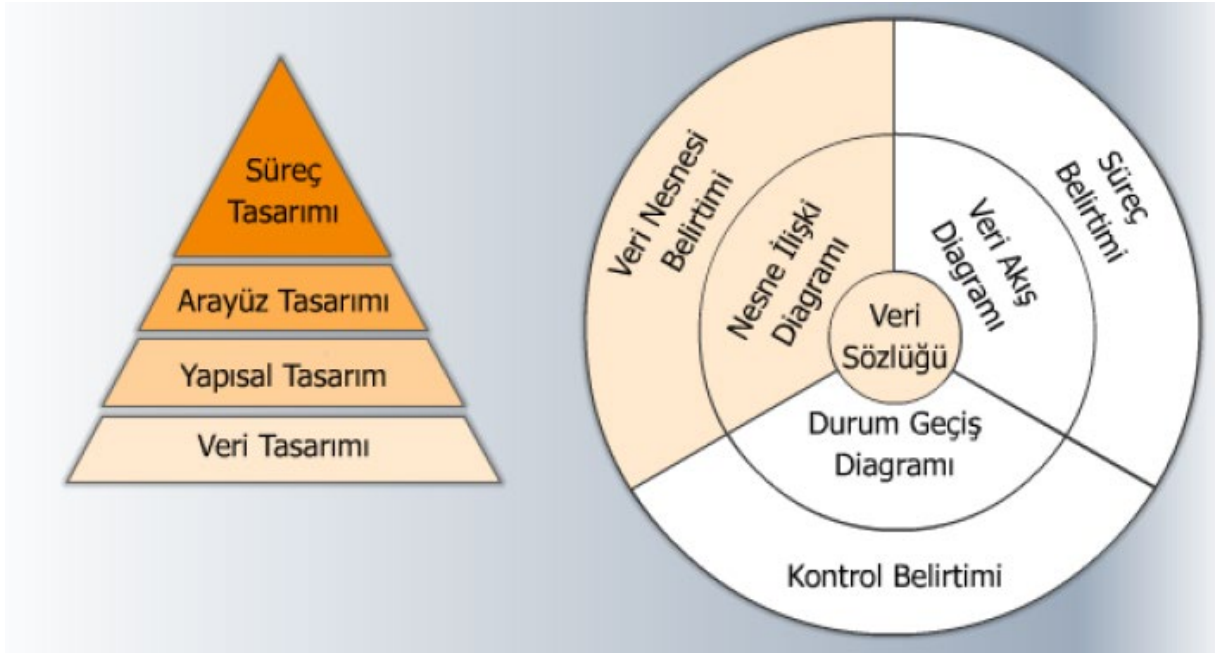
2.Süreç tasarımı bittikten sonra sıra Ara yüz tasarımına geldi ve arayüz tasarımı aşağıda görüldüğü adımlarla gerçekleştirildi.



3.Arayüz tasarımını aştıktan sonra sora yapısal tasarıma geldi. Yapısal tasarımda izlenen yollar aşağıdaki gibi oldu.

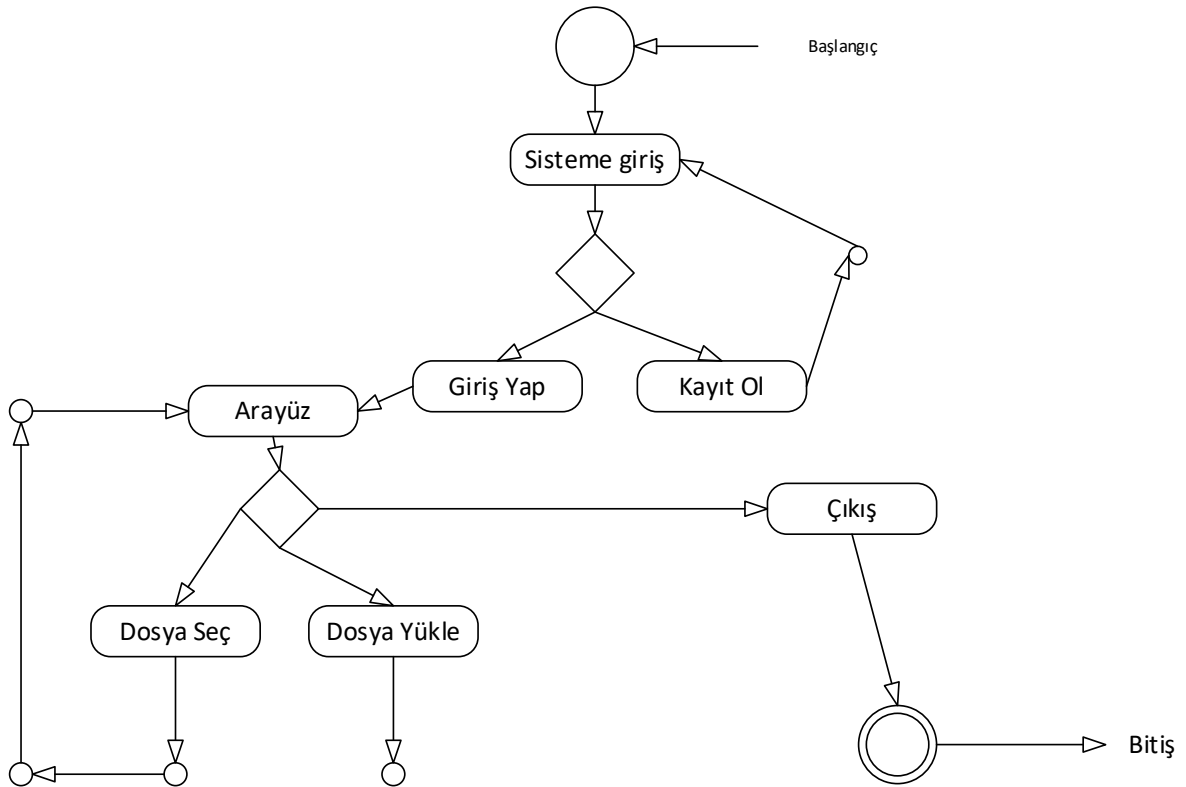


4.Yapısal tasarımı da yapıp Veri tasarımına geçildi ve şu adımlar izlendi.



Şekiller 4.1.1 Tasarım Aşamaları

#### 4.1.2 Sistem Mimarisi



### 4.1.3 Dış Arabirimler

#### *4.1.4.1 Kullanıcı Arabirimleri*

Kullanıcı arabirimlerin ilk başında sistem giriş ekranı bulunacak. Kullanıcı girişi bu arabirimde amaçlanmıştır. Kullanıcı bilgilerini girerek bu sisteme giriş yapmış olacaktır.

#### *4.1.4.2 Veri Arabirimleri*

#### *4.1.4.3 Diğer Sistemlerle Arabirimler*

Şuan tam bilgi sahibi olmadığımız için bu arabirim kullanılmayacaktır.

### 4.1.4 Veri Modeli

### 4.1.5 Testler

Genel hatlarıyla testlerimiz iki aşamada gerçekleştirilecek. Bilinen adıyla pilot bölge

Uygulaması yapılacak.

Alfa Aşaması: Sistemin geliştirildiği yerde kullanıcıların gelecek katkıda bulunması sistemi

Test etmesi ile yapılacak.

Beta Aşaması: Kullanıcı, geliştirilen sistemi kendi yerleşkesinde, bir gözetmen eşliğinde

Yapılacak.

### 4.1.6 Performans



Sistemin performansını etkileyen faktörlerin test verileri değerlendirilecek

Sistemin Tasarıma Uygunluk Performansı;

Tasarımı yapılan sistemin stabilizesi ve işleyiş performansı değerlendirilecek.

Veri Yapısının Sistemle Performansı;

Veri yapısının sistemle stabilizesi ve çalışma zamanındaki uyumluluk düzeyindeki

Performansı değerlendirilecek.

## 4.2 Veri Tasarımı

### 4.2.1 Tablo tanımları

Veri tipi olarak integer(int) kullanılmasının amacı sayısal değerleri almaktan ötürüdür. String olarak kullanılan değerler kelime içeren değerleri tutacağından ötürü kullanıldı.

### 4.2.2 Tablo- İlişki Şemaları

### 4.2.3 Veri Tanımları

## 4.3 Süreç Tasarımı

### 4.3.1 Genel Tasarım

Genel olarak tasarımda ilk önce veri tabanı modeli oluşturuldu. Ardından giriş modülü onun ardından yönetici modülleri ve en sonda kullanıcı ara yüzü oluşturduk.


### 4.3.2 Modüller

#### 4.3.2.1 Giriş Modülü


##### 4.3.2.1.1 İşlev


Kullanıcının sisteme müdahale edece bileceği ekrana erişmesi için aşması gereken bir modüldür.

##### 4.3.2.1.2 Kullanıcı Arabirimi




**Giriş Yap**

 Kullanıcı Adı

 Şifre

☒ Beni Hatırla



[Şifremi Unuttum](#)

#### 4.3.2.1.3 Modül Tanımı

Kullanıcı tarafından oluşturulan çok gizli güvenli kullanıcı adı ve şifre oluşturulur.

#### 4.3.2.1.4 Modül iç Tasarımı



Kullanılacak değişkenler:

**private Connection con:** Veri tabanı için bağlantı nesnesi.

Private PreparedStatement preparedStatement : Veri tabanı için işlem sıralaması

Private String url: Veri tabanının bulunduğu yol.

#### 4.3.2.2 Kullanıcı Modülü

#### 4.3.3 Kullanıcı Profilleri

**KULLANICI:** Ara yüzde mevcut olan her objeye erişebilir.

**ADMIN:** Ara yüz ve Arka plandaki bütün işlemlere erişebilir.

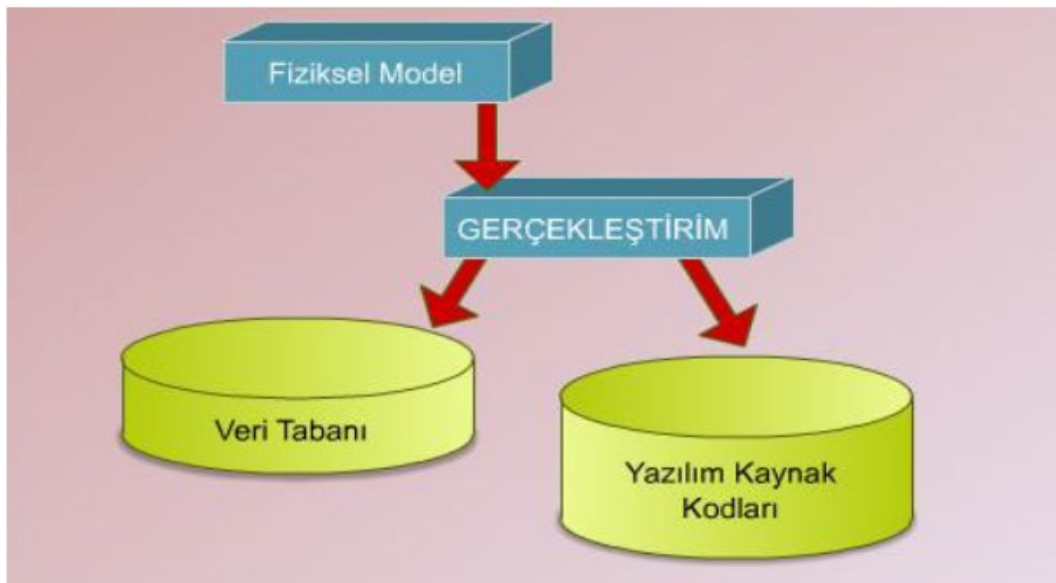
#### 4.3.4 Entegrasyon ve Test Gereksinimleri

Sistemimizin daha öncede belirttiğimiz gibi bir test ekibine ihtiyaç vardır.

### 5. SİSTEM GERÇEKLEŞTİRİMİ

#### 5.1. Giriş

Gerçekleştirim çalışması, tasarım sonucu üretilen süreç ve veri tabanının fiziksel yapısını içeren fiziksel modelin bilgisayar ortamında çalışan yazılım biçimine dönüştürülmesi çalışmalarını içerir. Yazılımın geliştirilmesi için her şeyden önce belirli bir yazılım geliştirme ortamının seçilmesi gerekmektedir.



## Şekil 5.1 Sistem Gerçekleştirim Modeli

### 5.2. Yazılım Geliştirme Ortamları

Yazılım geliştirme ortamı, tasarım sonunda üretilen fiziksel modelin, bilgisayar ortamında çalıştırılabilmesi için gerekli olan:

- Programlama Dili
- Veri Tabanı Yönetim Sistemi
- Hazır Program Kitapçıkları

CASE Araçları belirlendi ve yazılım geliştirme ortamı hazırlandı.

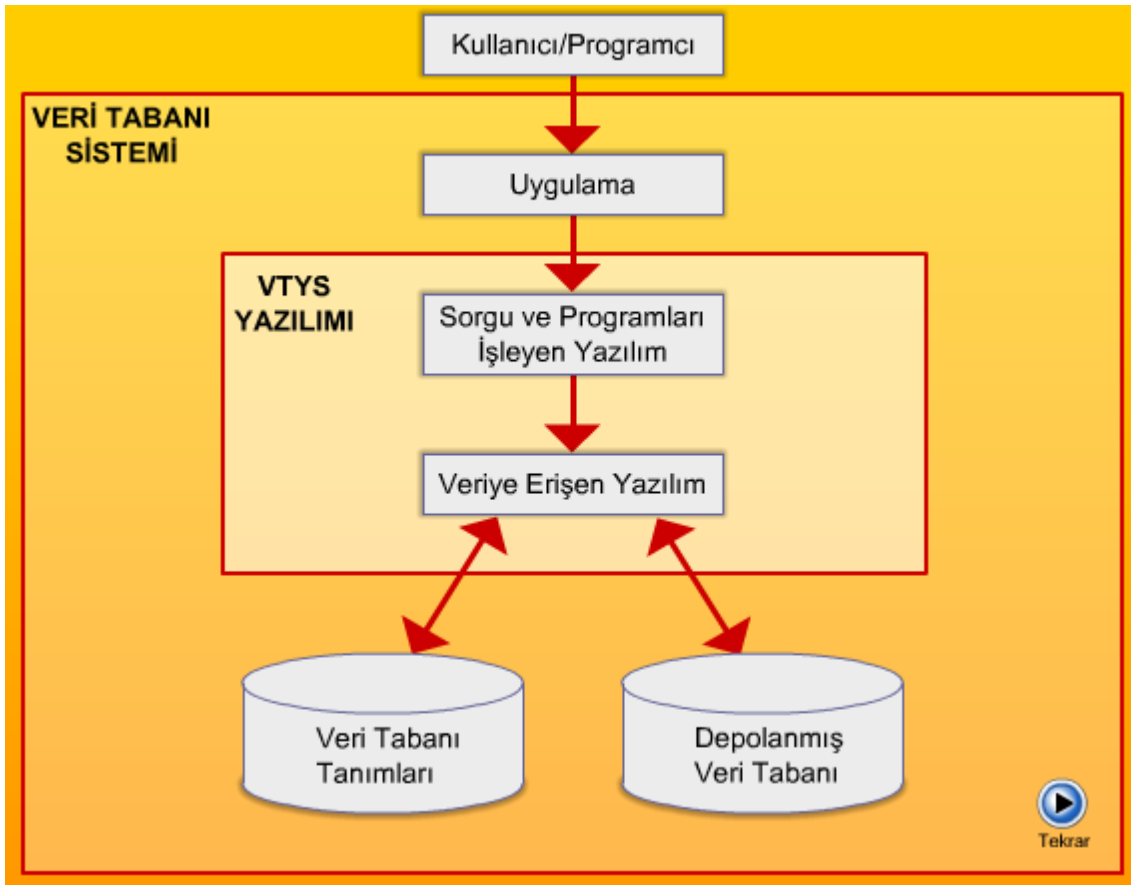
#### 5.2.1 Programlama Dilleri

Sistemde kullanılan başlıca programlama dillerini daha öncelerde de yazmıştık fakat burada bir kez daha dile getirelim. Kendi içlerinde sınıflandırılan bu diller arasından bizim seçtiğimiz sınıf şüphesiz ki veri işleme yoğunluklu uygulamalarda kullanılacak dillerden olacaktır. Bu yüzden daha çok görsel programlamaya önem verilecek kullandığımız diller arasında, SQL, JAVA başlıcalarıdır.

### 5.2.2 Veri Tabanı Yönetim Sistemleri

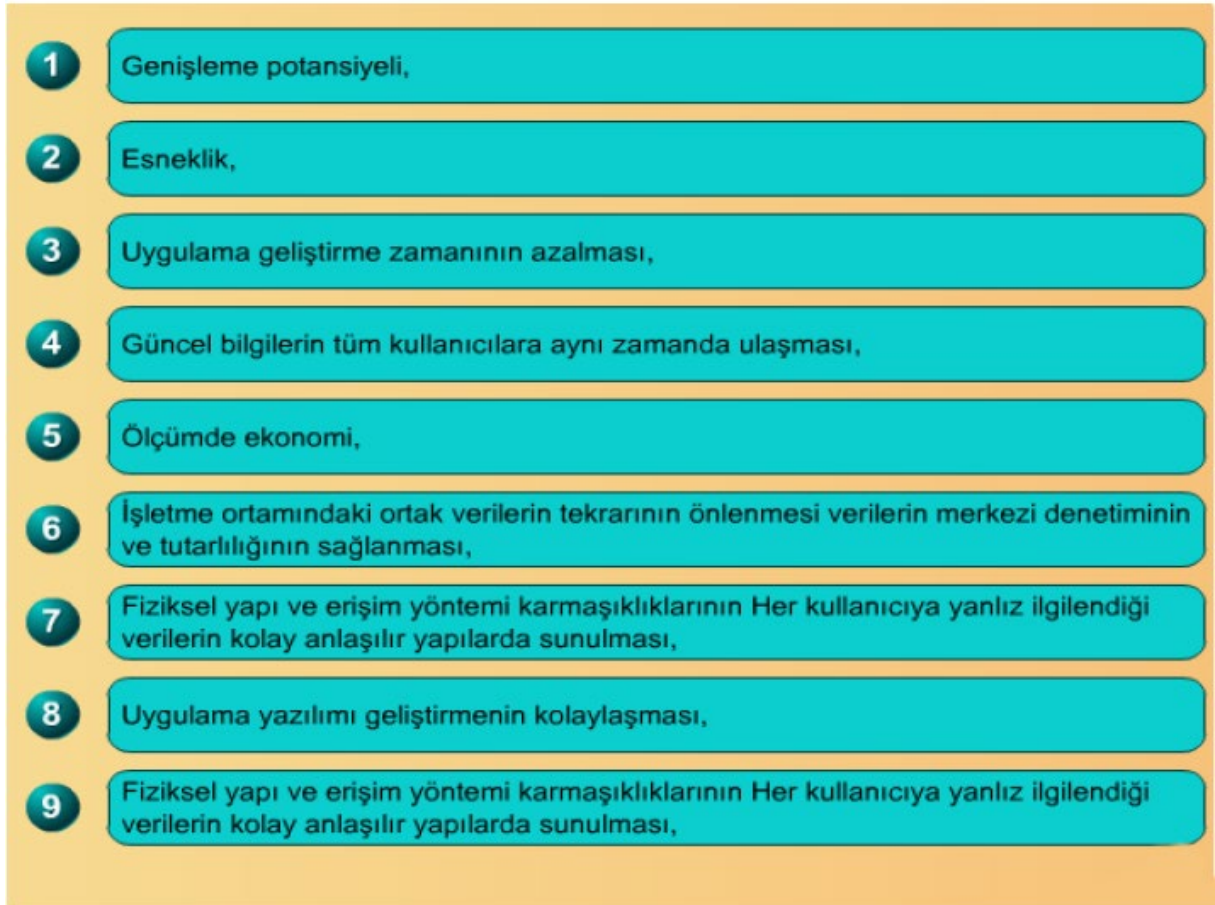
Veri tabanı tanımları ve Depolanmış veri tabanı ise SQL ile çözümleniyor.

Veri tabanı yönetiminde kullandığımız hiyerarşiyi bir göstermek gerekirse:



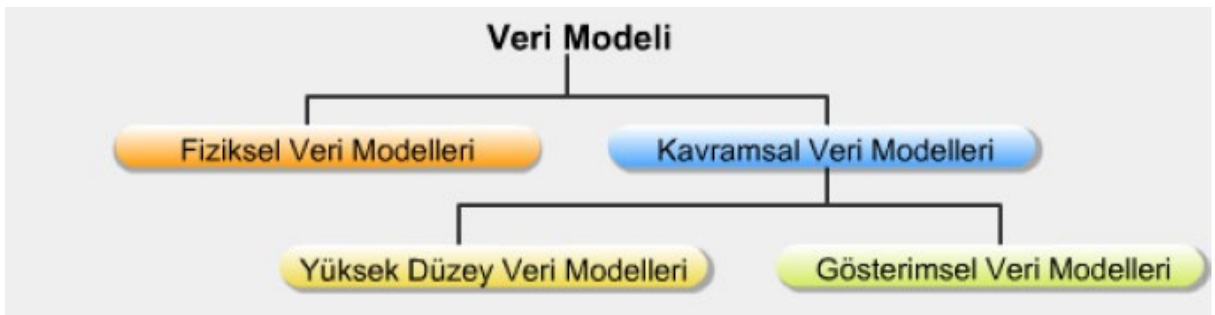
Şekil 5.2.2 Veri Tabanı Sistemi

#### 5.2.2.1 VTYS Kullanımının Ek Yararları

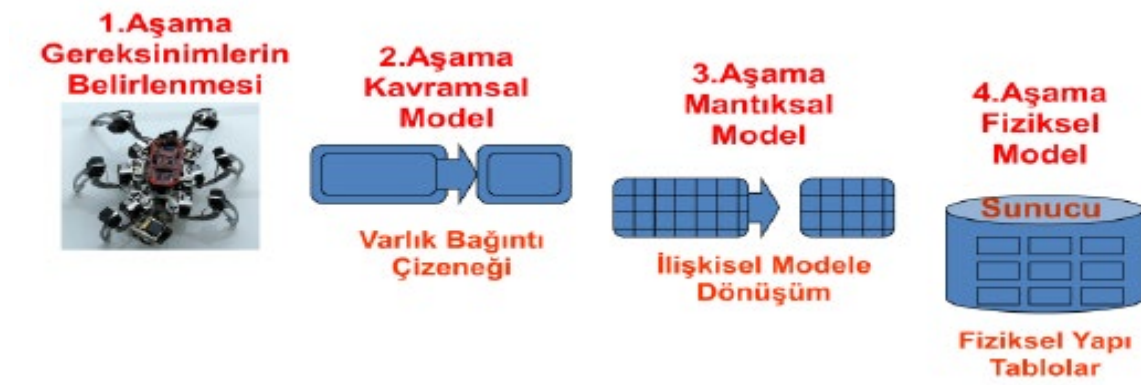


Şekil 5.2.2.1 VTYS Ek Yararları

#### 5.2.2.2 Veri Modelleri



Fiziksel Veri Modelinde verilerin Mysql de tablolar içindeki alanlarda saklanacağı ve birbirleriyle ilişki içinde olduğunu söyleyebiliriz. Kavramsal Veri Modelini iki ana başlık altında inceleyebiliriz.



Şekil 5.2.2.2 Şema Mimarisi

Üçlü şema mimarisinde görülen yapıların, kullanıcı gereksinimlerinden yola çıkılarak aşamalı bir şekilde fiziksel olarak gerçekleştirilmesidir.

### 1. Gereksinimlerin belirlenmesi

- Veri tipleri
- Veri grupları
- Veriler ile ilgili kurallar
- Veriler üzerinde yapılması gereken işlemler

### 2. Kavramsal Model

Kullanıcıdan elde edilecek gereksinimler ile ilgili bir analiz çalışmasının yapılması ve birbiriyle bağlantılı verilerin gruplanarak bir düzenleme içinde modellenmesi gerekmektedir. Bu modeli grafiksel olarak varlık bağıntı seçenekleri ile gösteririz.

### 3. Mantıksal Model

Veri tabanı tasarımlarımızın ilişkisel veri tabanı modelinde tablolar ile ifade edilebilmesi için yapılması gereken dönüşümü içerir.

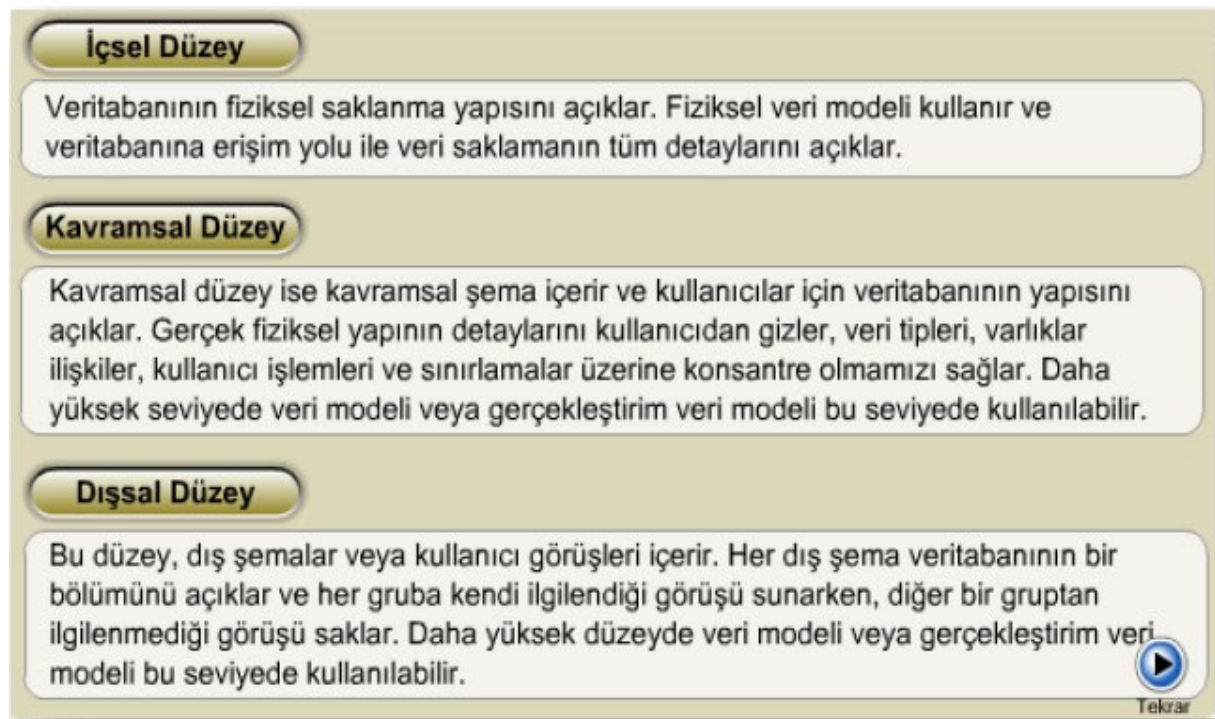
### 4. Fiziksel Model

Fiziksel olarak sistemin kurulması sağlanır. Kullanılacak VTYS ile ilgili ilk temas burada kurulur.

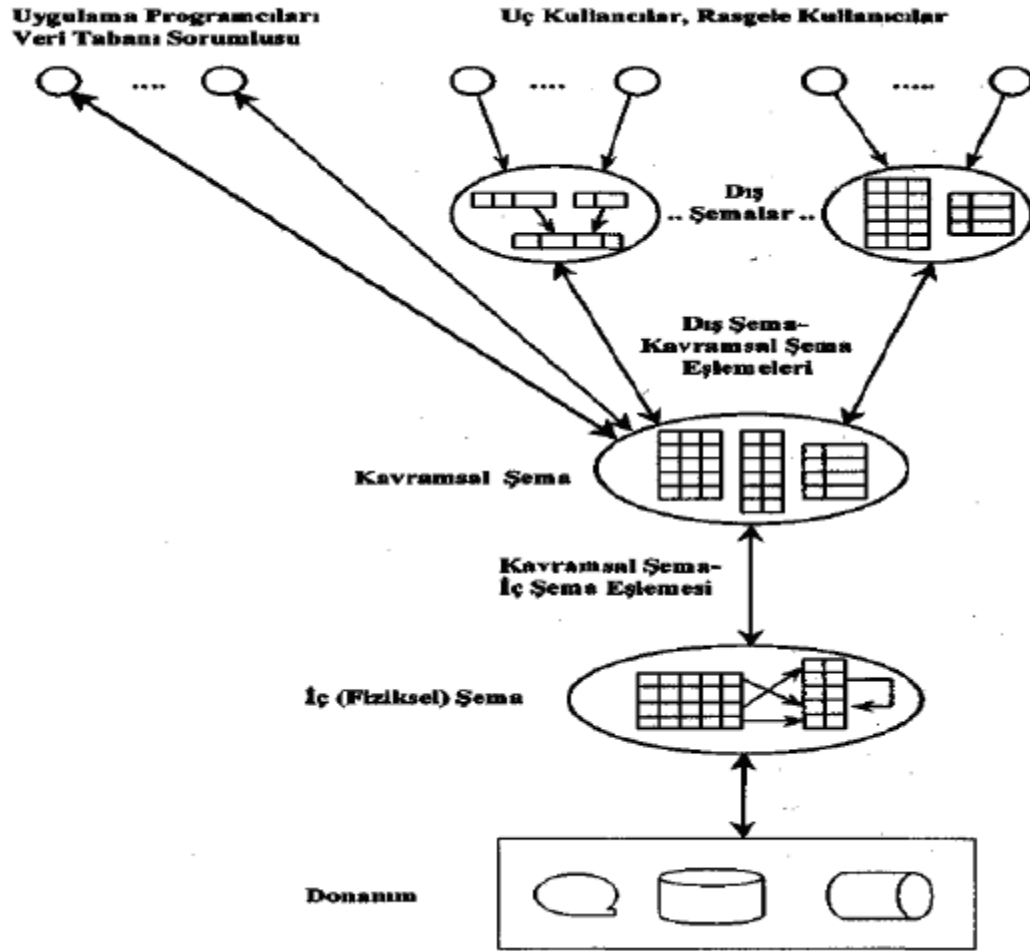
Herhangi bir veri modelinde veri tabanının tanımlanması ile kendisini ayırmak önemlidir. Veri tabanının tanımlamaları veri tabanı şeması veya meta-veri olarak adlandırılır. Veri tabanı şeması, tasarım sırasında belirtilir ve sıkça değişmesi beklenmez. Pek çok veri modeli şemaları, diyagramlar halinde göstermek için belli gösterim biçimlerine sahiptir. Diyagramlar her kayıt tipinin yapısını gösterir fakat kaydın gerçek örneğini göstermez.

#### 5.2.2.4 VTYS Mimarisi

VTYS mimarisini üç başlık altında ele aldık bunları özetlemek gerekir ise:







Şekil 5.2.2.4 Veri Tabanı Şemaları

#### 5.2.2.5 Veri tabanı Dilleri ve Arabirimleri

Sistemimizde veri tabanı dili olarak SQL kullanıldı. Henüz ilk örnek aşamasında olan sistemimiz için MYSQL arabirimini yeterli gördük.

<b>Veri Tanımlama Dili (VTD)</b>	Kavramsal şemaları tanımlamak üzere veritabanı yönetici ve tasarımcısı tarafından kullanılır
<b>Saklama Tanımlama Dili (STD)</b>	İşsel şemayı tanımlamak için kullanılır.
<b>Görüş Tanımlama Dili (GTD)</b>	Görüş tanımlama dili kullanıcı görüşlerini tanımlamak ve kavramsal şemaya dönüştürmek amacıyla kullanılır.
<b>Veri İşleme Dili (VID)</b>	Veri işleme dilidir. Veri tabanı oluşturulduktan sonra Veri tabanına veri eklemek, değiştirmek, silmek veya eklenmiş veriyi getirmek amacıyla kullanılır.

Şekil 5.2.2.5 Veri tabanı Dilleri ve Arabirimleri

#### 5.2.2.6 Veri Tabanı Sistem Ortamı

Veri tabanı sistem ortamında phpMyAdmin bizim kahrımızı çekti. Tüm Yükleme, yedekleme, performans ölçme, sıralama, veri sıkıştırma ve benzeri fonksiyonları yerine getirmek amacıyla bu ortamı kullandık.

#### 5.2.2.7 VTYS'nin Sınıflandırılması

En fazla kullanılan veri modelleri ilişkisel, ağ, hiyerarşik, nesne-yönelimli ve kavramsal modellerdir. Bizim Kullandığımız ise ilişkisel veri modelidir.

#### 5.2.2.8 Hazır Program Kütüphane Dosyaları

Zaten entegre olduğu için böyle bir şeye ihtiyaç duymadık.

#### 5.2.2.9 CASE Araç ve Ortamları

Case araçları olarak ise Microsoft'un Visio Project ürünlerini kullandık.

### 5.3. Kodlama Stili

Kendimize has kodlama biçimini kullandık herhangi bir hazır düzene bağlı kalmadık. Bakım programcımıza da aynı stil üzerine eğitim verdik ve sorunları ortadan kaldırdık.

#### 5.3.1 Açıklama Satırları

Açıklama satırları her metod ve karmaşık yerlerde konumlandırıldı.

#### 5.3.2 Kod Biçimlemesi

Kod biçimlemesine değinmek gerekirse alt alta oluşan kodlarda tabi indexleri kullandık ve iç içe bir biçimde hiyerarşi oluşturduk.

#### 5.3.3 Anlamlı İsimlendirme

Sistem kodlamasının genel yapısında kullanılan değişkenlerin veri tabanında karşılığı varsa önce “tablodaki işlev” şeklinde bir anlamlı isimlendirme yaptık.

#### 5.3.4 Yapısal Programlama Yapıları

Genel olarak 3 başlıkta incelersek:

- **Ardışık işlem yapıları:** Bu tür yapılarda genellikle fonksiyon, altprogram ve buna benzer tekrarlı yapıları tek bir seferde çözdük.
- **Koşullu işlem yapıları:** Bu yapıları ise neredeyse programın tamamında kullandık karşılaştırma yapılan her yerde bunlara yer verildi.
- **Döngü yapıları:** Tıpkı ardışık işlemler gibi alt alta birkaç satır yazıcığımıza tek bir döngüyle bu sorunların üstesinden geldik.

#### 5.4. Program Karmaşıklığı

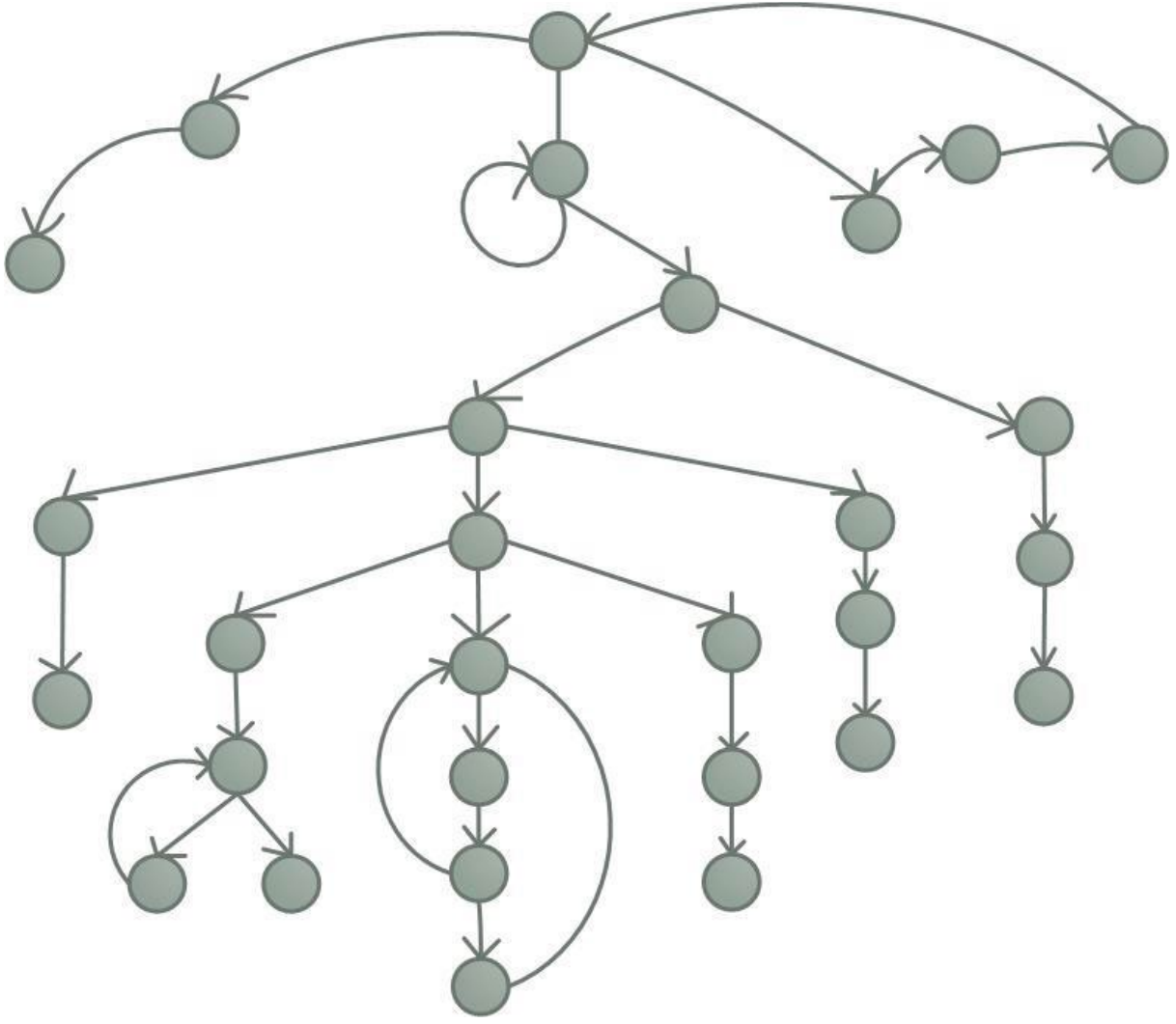
“Program karmaşıklığını ölçmek için birçok teorik model geliştirilmiştir. Bu modellerin en eskisi ve yol göstericisi McCabe karmaşıklık ölçütüdür. Bu bölümde bu ölçüt anlatılmaktadır. Söz konusu ölçüt 1976 yılında McCabe tarafından geliştirilmiştir. Bu konuda geliştirilen diğer ölçütlerin çoğu, bu ölçütten esinlenmiştir.

McCabe ölçütü, bir programda kullanılan "koşul" deyimlerinin program karmaşıklığını etkileyen en önemli unsur olduğu esasına dayanır ve iki aşamada uygulanır: “



Şekil 5.4Program Karmaşıklığı

##### 5.4.1 Programın Çizge Biçimine Dönüştürülmesi



Şekil 5.4.1 Programın Çizgi Hali

#### 5.4.2 McCabe Karmaşıklık Ölçütü Hesaplama

**k = 13** Kenar sayısı **d = 9** Düğüm sayısı **p = 1** Bileşen sayısı

**$V(G)=k-d+2p$  (Formülüyle bulunur)**

#### 5.5. Olağan Dışı Durum Çözümleme

Olağan dışı durum, bir programın çalışmasının, geçersiz ya da yanlış veri oluşumu ya da başka nedenlerle istenmeyen bir biçimde sonlanmasına neden olan durum olarak tanımlanmaktadır.

### 5.5.1 Olağandışı Durum Tanımları

Olağandışı gelişen durumlarda try-catch blokları devreye girecek ve program kırılmadan çalışmasına devam edebilecek şekilde tasarladık.

### 5.5.2 Farklı Olağandışı Durum Çözümleme Yaklaşımları

Tüm olağan dışı durumlarda program kırılmadan hata mesajlarıyla tekrar başa dönecek şekilde tasarladık.



Şekil 5.5.2 Olağan Dışı Halde Yapılacaklar

### 5.6. Kod Gözden Geçirme

Hiç kimse, önceki sürümlerini gözden geçirmeden ve incelemeyen okunabilir bir program yazamaz. Hiçbir yazı editörün onayını almadan basılamayacağı gibi hiçbir program da

incelenmeden, gözden geçirilmeden işleme alınmamalıdır. Kod gözden geçirme ile program sınaıa işlemlerini birbirinden ayırmak gerekir.

Program sınaıa, programın işlemini sırasında ortaya çıkabilecek yanlış ya da hataları yakalamak amacıyla yapılır. Kod gözden geçirme işlemini ise, programın kaynak kodu üzerinde yapılan bir incelemedir. Kod gözden geçirmelerinde program hatalarının %3-5 oranındaki kesimi yakalanabilmektedir. Eğer programı yazan kişi, yazdığı programın hemen sonra bir "kod inceleme" sürecine girdi olacağını bilerek program yazdığında daha etkin, az hatalı ve okunabilir programlar elde edilebilmektedir.

#### 5.6.1 Gözden Geçirme Sürecinin Düzenlenmesi

Gözden geçirme sürecinin temel özellikleri;

- Hataların bulunması, ancak düzeltilmemesi hedeflenir,
- Olabildiğince küçük bir grup tarafından yapılmalıdır. En iyi durum deneyimli bir inceleyci kullanılmasıdır. Birden fazla kişi gerektiğinde, bu kişilerin, ileride program bakımı yapacak ekipten seçilmesinde yarar vardır.
- Kalite çalışmalarının bir parçası olarak ele alınmalı ve sonuçlar düzenli ve belirlenen bir biçimde saklanmalıdır. Biçiminde özetlenebilir. Burada yanıtı aranan temel soru, programın yazıldığı gibi çalışıp çalışmayacağını belirlemesidir.

#### 5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular

Bir program incelenirken, programın her bir öbeğı (yordam ya da işlev) aşağıdaki soruların yanıtları aranır. Bu sorulara ek sorular eklenebilir. Bazı soruların yanıtlarının "hayır" olması programın reddedileceğı anlamına gelmemelidir.

##### 5.6.2.1 Öbek Ara yüzü

Oluşturduğumuz öbekleri test etmek için belli sorular sorduk bu sorular:

- Her öbek tek bir işlevsel amacı yerine getiriyor mu?

- Öbek adı, işlevini açıklayacak biçimde anlamlı olarak verilmiş mi?
- Öbek tek giriş ve tek çıkışlı mı?
- Öbek eğer bir işlev ise, parametrelerinin değerini değiştiriyor mu?

Şeklinde oldu.

#### 5.6.2.2 Giriş Açıklamaları

Oluşturduğumuz giriş açıklamalarını test etmek için belli sorular sorduk bu sorular:

- Öbek, doğru biçimde giriş açıklama satırları içeriyor mu?
- Giriş açıklama satırları, öbeğin amacını açıklıyor mu?
- Giriş açıklama satırları, parametreleri, küresel değişkenleri içeren girdileri ve kütükleri tanıtıyor mu?
- Giriş açıklama satırları, çıktıları (parametre, kütük vb.) ve hata iletilerini tanımlıyor mu?
- Giriş açıklama satırları, öbeğin algoritma tanımını içeriyor mu?
- Giriş açıklama satırları, öbekte yapılan değişikliklere ilişkin tanımlamaları içeriyor mu?
- Giriş açıklama satırları, öbekteki olağan dışı durumları tanımlıyor mu?
- Giriş açıklama satırları, Öbeği yazan kişi ve yazıldığı tarih ile ilgili bilgileri içeriyor mu?
- Her paragrafı açıklayan kısa açıklamalar var mı?

Şeklinde oldu.

#### 5.6.2.3 Veri Kullanımı

Oluşturduğumuz veri kullanımlarını test etmek için belli sorular sorduk bu sorular:

- İşlevsel olarak ilintili bulunan veri elemanları uygun bir mantıksal veri yapısı içinde gruplanmış mı?
- Değişken adları, işlevlerini yansıtacak biçimde anlamlı mı?



- Değişkenlerin kullanımları arasındaki uzaklık anlamlı mı?
- Her değişken tek bir amaçla mı kullanılıyor?
- Dizin değişkenleri kullanıldıkları dizinin sınırları içerisinde mi tanımlanmış?
- Tanımlanan her gösterge değişkeni için bellek ataması yapılmış mı?

Şeklinde oldu.

#### 5.6.2.4 Öbeğin Düzenlenişi

- Modüller birleşimi uyumlumu?
- Modüller arası veri aktarımları sağlanıyor mu?
- Bütün modüller birleştiğinde sistem çalışıyor mu? Gözden geçirme sırasında referans alınacak sorular olacaktır.

#### 5.6.2.5 Sunuş

Artık son kısma gelindiğinde ise şu sorular soruldu:

- Her satır, en fazla bir deyim içeriyor mu?
- Bir deyim birden fazla satıra taşması durumunda, bölünme anlaşılabilirliği kolaylaştıracak biçimde anlamlı mı?
- Koşullu deyimlerde kullanılan mantıksal işlemler yalın mı?
- Bütün deyimlerde, karmaşıklığı azaltacak şekilde parantezler kullanılmış mı?
- Bütün deyimler, belirlenen program stiline uygun olarak yazılmış mı?
- Öbek yapısı içerisinde akıllı "programlama hileleri" kullanılmış mı?

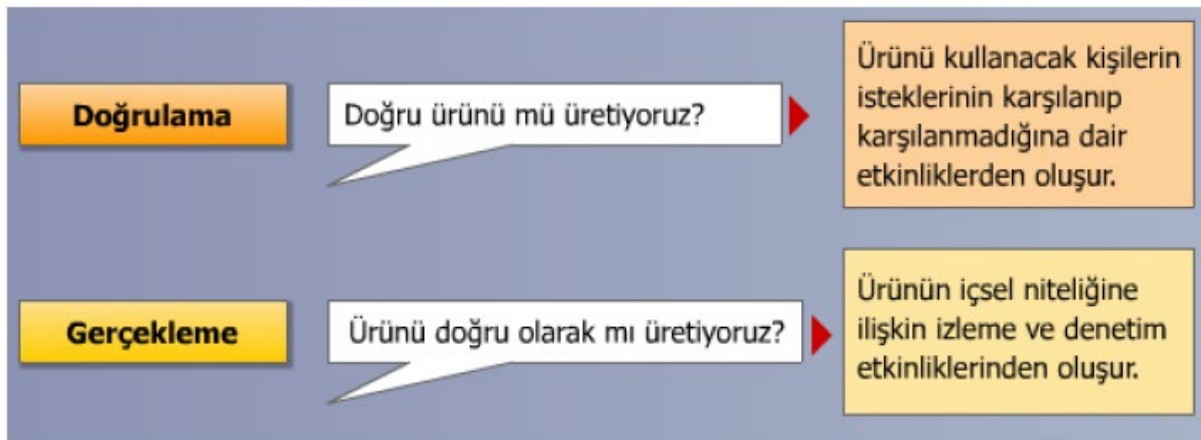
## 6. DOĞRULAMA VE GEÇERLEME

### 6.1. Giriş

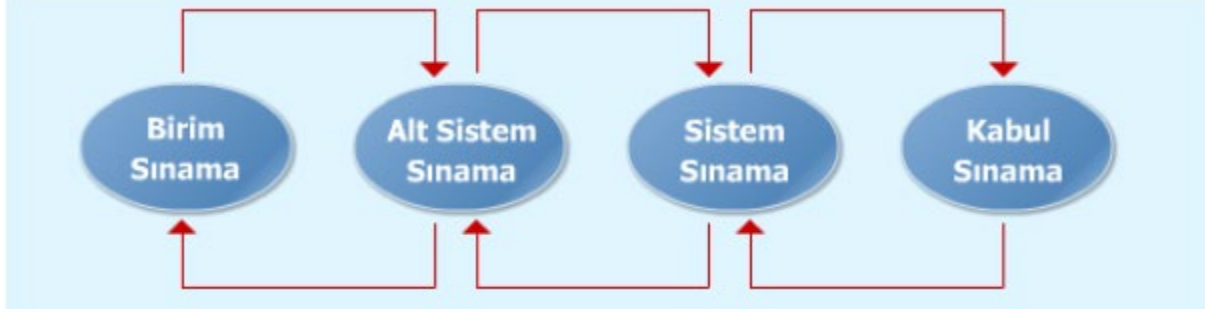
Geliştirilecek bilgi sistemi yazılımının doğrulanması ve geçerlemesi, üretim süreci boyunca süren etkinliklerden oluşur. Söz konusu etkinlikler:

- Yazılım belirtilerinin ve proje yaşam sürecindeki her bir etkinlik sonunda alınan çıktıların, tamam, doğru, açık ve önceki belirtileri tutarlı olarak betimler durumda olduğunun doğrulanması.
- Proje süresince her bir etkinlik ürününün teknik yeterliliğinin değerlendirilmesi ve uygun çözüm elde edilene kadar aktivitenin tekrarına sebep olması.
- Projenin bir aşaması süresince geliştirilen anahtar belirtilerin önceki belirtilerle karşılaştırılması.

Yazılım ürünlerinin tüm uygulanabilir gerekleri sağladığının gerçekleştirilmesi için sınamaların hazırlanıp yürütülmesi biçiminde özetlenebilir.



## 6.2. Sınama Kavramları



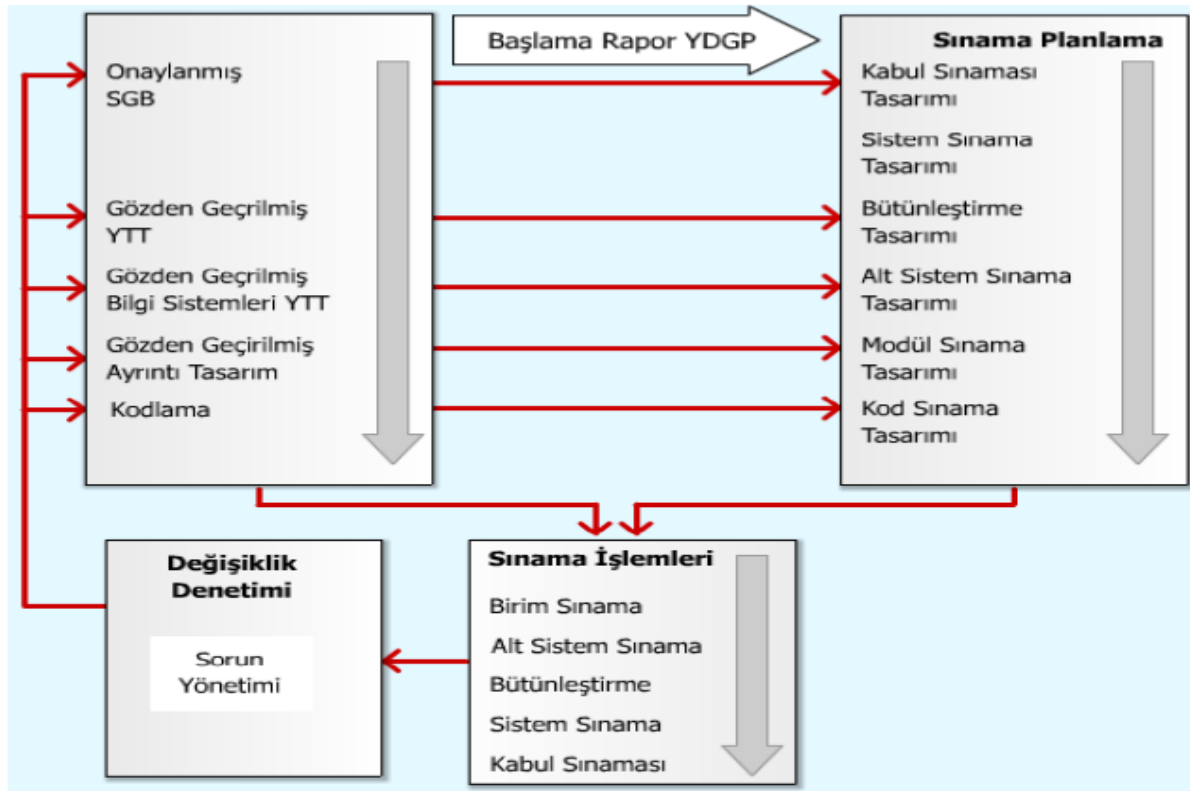
**Birim Sınama:** Sistemin birimleri sılandı ve sonuçları çıkartıldı.

**Alt Sistem Sınama:** Birimlerin birleşmesiyle modüller oluşturulup bunların kendi içinde sınaması yapıldı. Genel olarak ara yüzde ki eksiklikler giderildi.

**Sistem Sınama:** Sistemin bütün olarak sınanması yapıldı ve programın eksiksiz olduğu onaylandı.

**Kabul Sınama:** Sistem prototipten çıkartılıp gerçek veriler girildi ve sorunsuz olduğu bir kez daha onaylandı.

## 6.3. Doğrulama ve Geçerleme Yaşam Döngüsü



Şekil 6.3 Yaşam Döngüsü

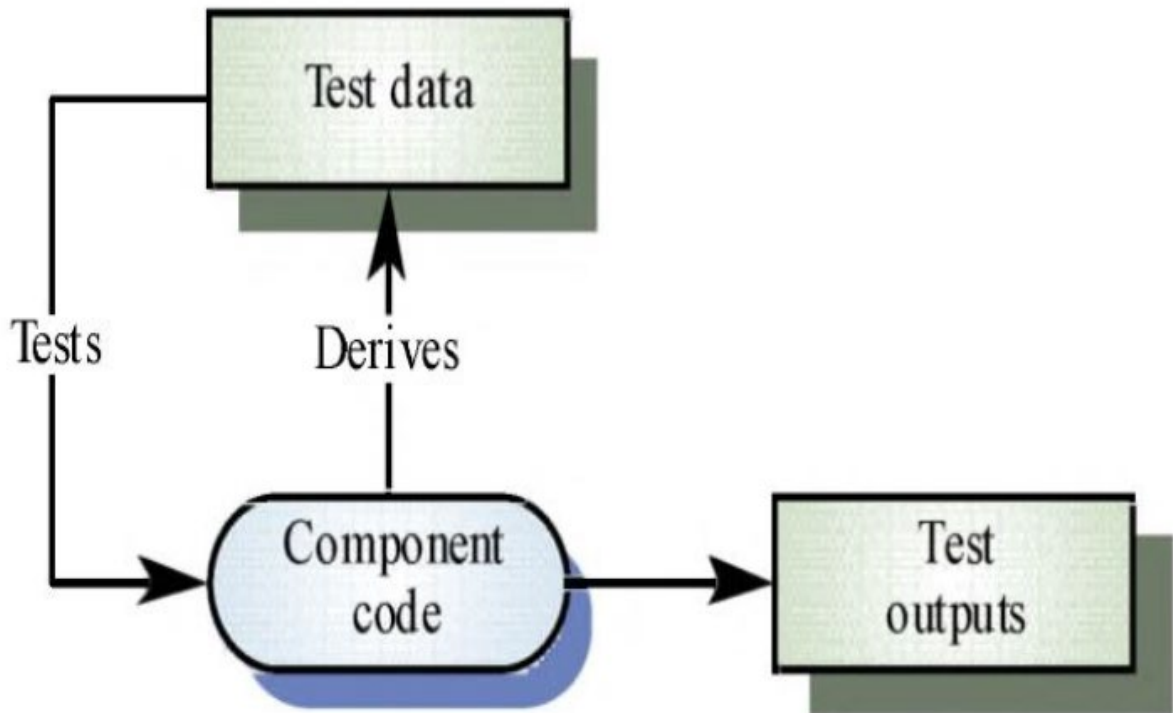
#### 6.4. Sınama Yöntemleri

Sınama işlemi, geliştirmeyi izleyen bir düzeltme görevi olmak ile sınırlı değildir. Bir "sonra" operasyonu olmaktan çok, geliştirme öncesinde planlanan ve tasarımı yapılması gereken bir çaba türüdür.

##### 6.4.1 Beyaz Kutu Sınaması

Denetimler arasında:

- Bütün bağımsız yolların en azından bir kere sınanması,
- Bütün mantıksal karar noktalarında iki değişik karar için sınamaların yapılması,
- Bütün döngülerin sınır değerlerinde sınanması,
- İç veri yapılarının denenmesi yapıldı.

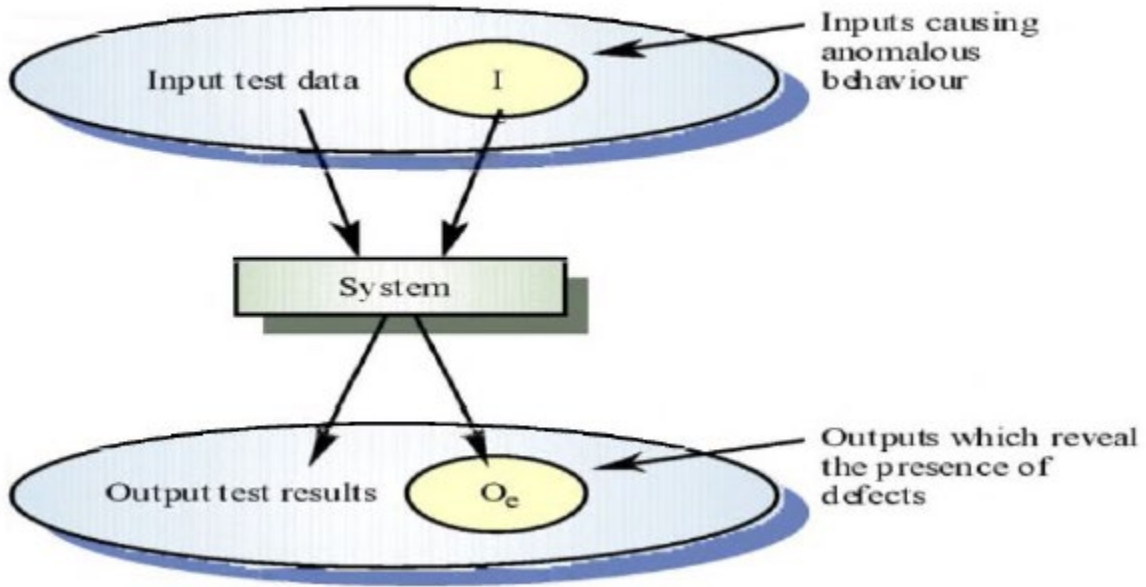


Şekil 6.4.1 Beyaz Kutu Sınaması

#### 6.4.2 Temel Yollar Sınaması

Sistemin tümüne yönelik işlevlerin doğru yürütüldüğüne testidir. Sistem şartnamesinin gerekleri incelenir.

- Eş değerlendirme bölme
- Uç değerler analizi
- Karar Tablosu
- Sonlu durum makinesi
- Belgelenmiş özelliklere göre test
- Rastgele Test
- Kullanım profili



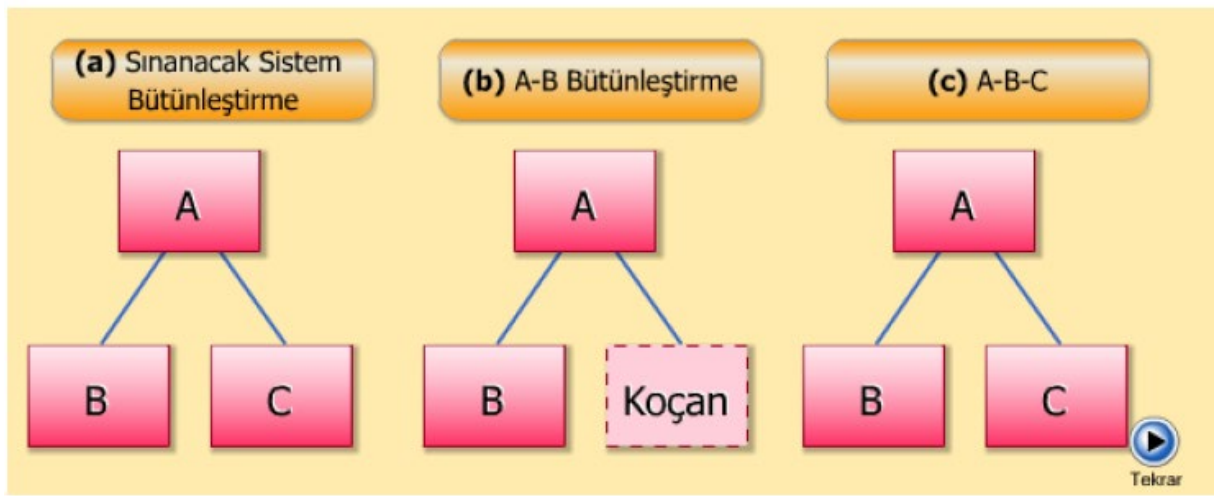
Şekil 6.4.2 Sınama Şekli

#### 6.5. Sınama ve Bütünleştirme Stratejileri

Genellikle sınama stratejisi, bütünleştirme stratejisi ile birlikte değerlendirilir. Ancak bazı sınama stratejileri bütünleştirme dışındaki tasaları hedefleyebilir. Örneğin, yukarıdan aşağı ve aşağıdan yukarı stratejileri bütünleştirme yöntemine bağımlıdır. Ancak işlem yolu ve gerilim sınamaları, sistemin olaylar karşısında değişik işlem sırtlandırmaları sonucunda ulaşacağı sonuçların doğruluğunu ve normal şartların üstünde zorlandığında dayanıklılık sınırını ortaya çıkarır.

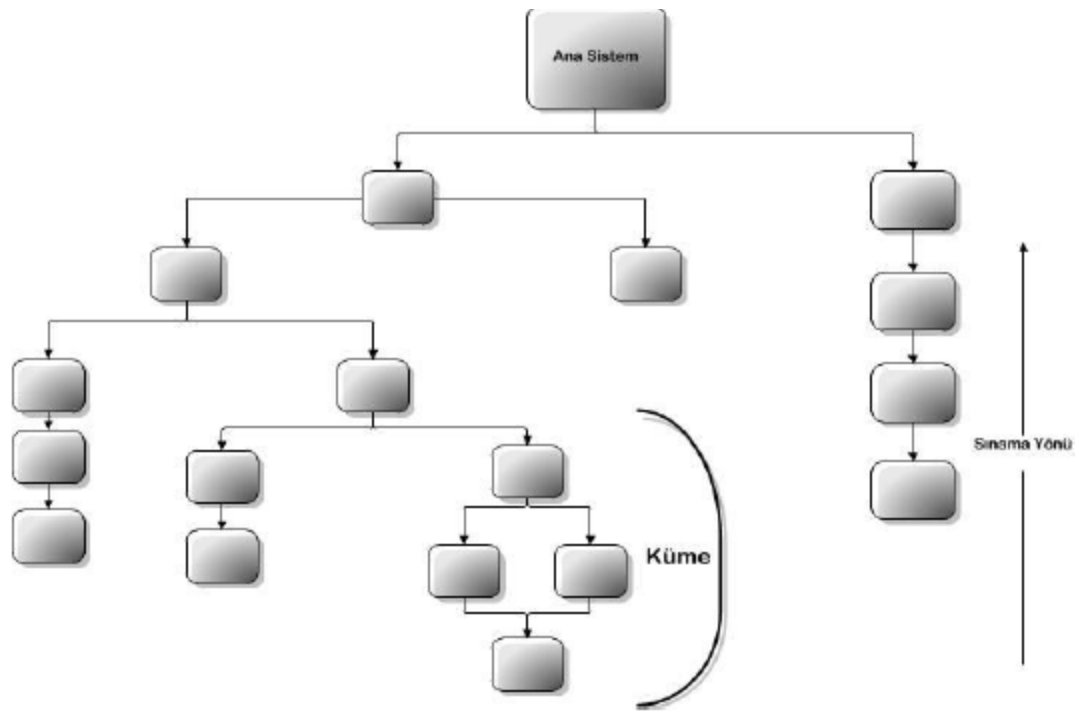
### 6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme

Yukarıdan aşağı bütünleştirmede, önce sistemin en üst düzeylerinin sınanması ve sonra aşağıya doğru olan düzeyleri, ilgili modüllerin takılarak sınanmaları söz konusudur. En üst noktadaki bileşen, bir birim/modül/alt sistem olarak sılandıktan sonra alt düzeye geçilmelidir. Ancak bu en üstteki bileşenin tam olarak sınanması için alttaki bileşenlerle olan bağlantılarının da çalışması gerekir. Genel hatlarıyla özetlemek gerekirse şu mantıkla sitem sınaması yapıldı.



### 6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme

Aşağıdan yukarı bütünleştirmede ise, önceki yöntemin tersine uygulama yapılır. Önce en alt düzeydeki işçi birimleri sınanır ve bir üstteki birimle sınama edilmesi gerektiğinde bu üst bileşen, bir 'sürücü' ile temsil edilir. Yine amaç, çalışmasa bile ara yüz oluşturacak ve alt bileşenin sınanmasını sağlayacak bir birim edinmektir. Fakat bu sınama sistemi kullanılmadı.



**Sorumluluklar:** Hangi kişilerin nelerden sorumlu olduğu ve test takım lideri bilgileri mutlaka raporda belirtilmelidir.

**Riskler ve Önlemler:** Test planında varsayılan ve olası yüksek riskli durumları belirtir ve bu durumların olması durumunda, etkilerinin en aza indirilebilmesi için alınması gereken önlemleri açıklar.

<b>Giriş</b>
Amaç Tanım ve Kısaltmalar Referanslar
<b>Sinama Yönetimi</b>
Sinama Konusu Sinama Etkinlikleri ve Zamanlama  Temel Sinama Etkinlikleri Destek Etkinlikler  Kaynaklar ve Sorumluluklar Personel ve Eğitim Gereksemeleri Sinama Yaklaşımı Riskler ve Çözümler Onaylar
<b>Sinama Ayrıntıları</b>
Sinanacak Sistemler Girdiler ve Çıktılar Sinamaya Başlanma Koşulları  Girdilerin Hazır Olması Ortam Koşulları Kaynak Koşulları

## 6.7. Sinama Belirtileri

Sinama belirtileri, bir sinama işleminin nasıl yapılacağına ilişkin ayrıntıları içerir



Bu ayrıtlar temel olarak:

- sinanan program modülü ya da modüllerinin adları,
- sinama türü, stratejisi (beyaz kutu, temel yollar vb.),
- sinama verileri,
- sinama senaryoları

türündeki bilgileri içerir.

Sinama verilerinin elle hazırlanması çoğu zaman kolay olmayabilir ve zaman alıcı olabilir. Bu durumda, otomatik sinama verisi üreten programlardan yararlanılabilir.

Sinama senaryoları, yeni sinama senaryosu üretebilmeye yardımcı olacak biçimde hazırlanmalıdır. Zira sinama belirtilmelerinin hazırlanmasındaki temel amaç, etkin sinama yapılması için bir rehber oluşturması

Sinama işlemi sonrasında bu belirtilmelere,

- sinamayı yapan,
- sinama tarihi,
- bulunan hatalar ve açıklamaları

türündeki bilgiler eklenerek sinama raporları oluşturulur.

Sinama raporları, sinama bitiminde imzalanır ve yüklenici ile iş sahibi arasında resmi belge niteliği oluşturur.

## 6.8. Yaşam Döngüsü Boyunca Sinama Etkinlikleri

Bütün bu etkinlikleri bir hiyerarşi altında incelemek gerekirse:

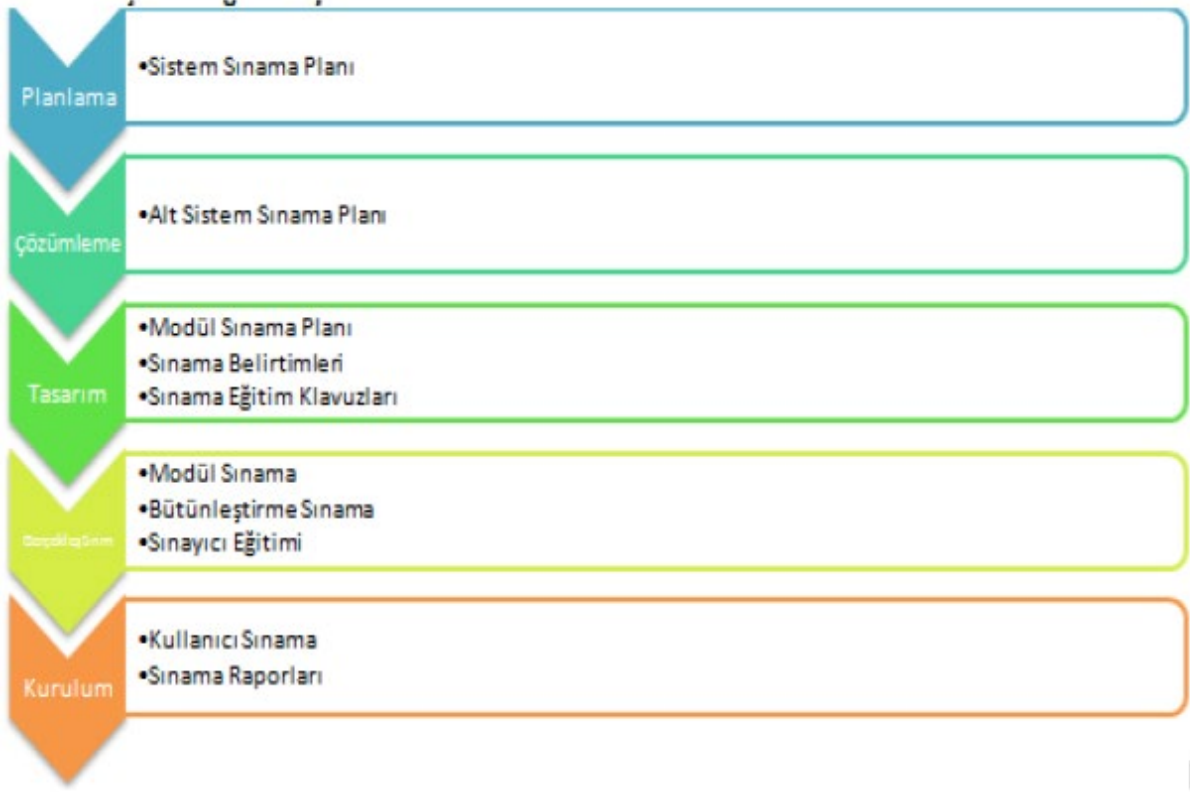
Planlama aşamasında genel planlama sinaması gerçekleştirilir. Bu olan tüm planların basit bir ön hazırlığı niteliğindedir.

Çözümleme aşamasında sinama planı alt sistemler bazında ayrıntılandırılır.

Tasarım aşamasında sinama plana detaylandırılır ve sinama belirtilmeleri oluşturulur. Bu oluşumlar daha sonra eğitim ve el kitabında kullanılır.

Gerçekleştirim aşamasında teknik sinamalar yapılır sinama raporları hazırlanır ve elle tutulur ilk testler yapılır.

Kurulum aşamasında sistemle ilgili son sinamalar yapılır ve sinama raporları hazırlanır.



Şekil 6.8 Sınama Etkinlikleri

Bu ayrıtlar temel olarak:

- Sınanan program modülü ya da modüllerinin adları,
- Sınama türü, stratejisi (beyaz kutu, temel yollar vb.),
- Sınama verileri,
- Sınama senaryoları

Türündeki bilgileri içerir.

Sınama sırasında bulunan her hata için, **değişiklik kontrol sistemine (DKS)**, "Yazılım Değişiklik İsteği" türünde bir kayıt girilir. Hatalar, DKS kayıtlarında aşağıdaki gibi gruplara ayrılabilir:

- Onulmaz Hatalar: BT projesinin gidişini bir ya da birden fazla aşama gerileten ya da düzeltilmesi mümkün olmayan hatalardır.
- Büyük Hatalar: Projenin kritik yolunu etkileyen ve önemli düzeltme gerektiren hatalardır.

- Küçük Hatalar: Projeyi engellemeyen ve giderilmesi az çaba gerektiren hatalardır
- Hatalar: Heceleme hatası gibi önemsiz hatalardır.

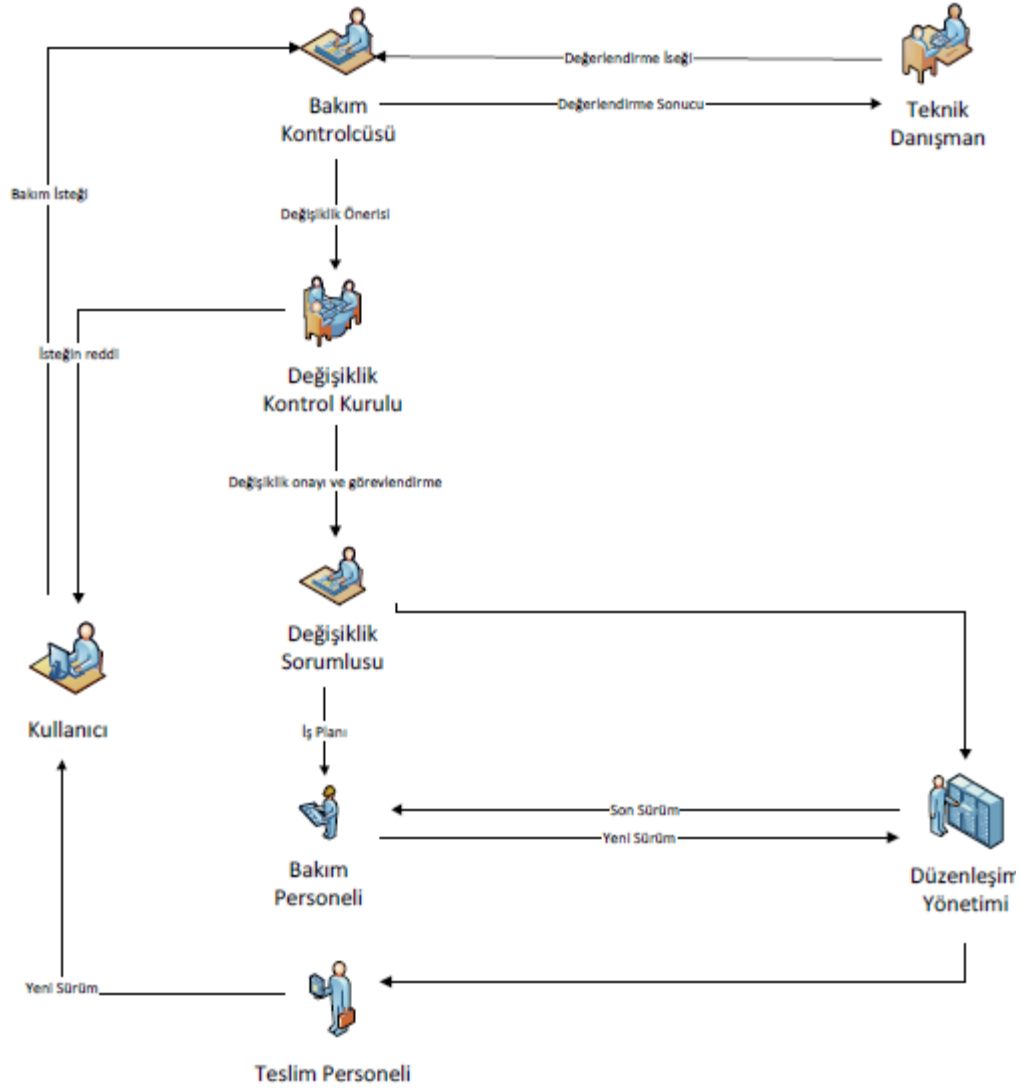


Gerçekleştirme Ve Doğrulama-Zaman Diyagramı			
Zaman	7.Hafta	8.Hafta	9.Hafta
Gerçekleştirme-Doğrulama			
Gerçekleştirme			
Doğrulama			

## 7. BAKIM

### 7.1 Giriş

Sistemin tasarımı bittikten sonra artık seçimden seçime sistemin bakıma sokulması gerekir daha öncede belirttiğimiz gibi sistem hassas ve hata kabul etmeyecek bir sistemden bahsediyoruz. Bakım bölümüne ilişkin yapılan açıklamalarda IEEE 1219-1998 standardı taban olarak alınmıştır.



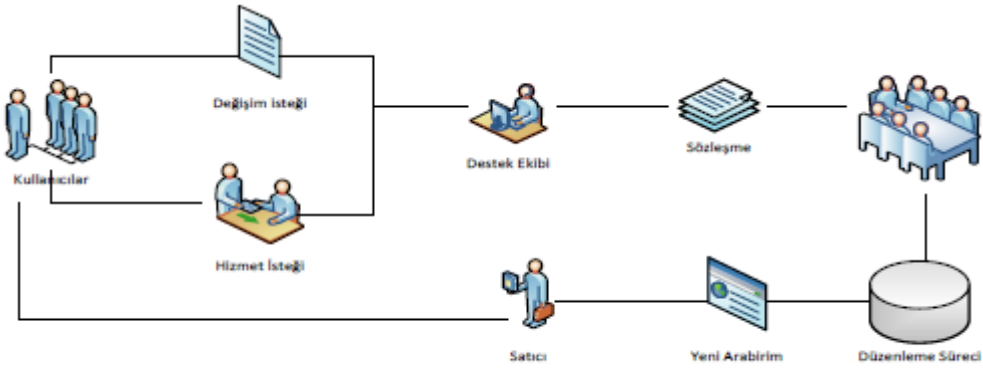
## 7.2 Kurulum

Sistem kurulumuna değinmek gerekirse devlet güvencesinde verilecek olan serverlara yüklenecek ve internet Ortamı mevcut olması gerekiyor.

### 7.3 Yerinde Destek Organizasyonu

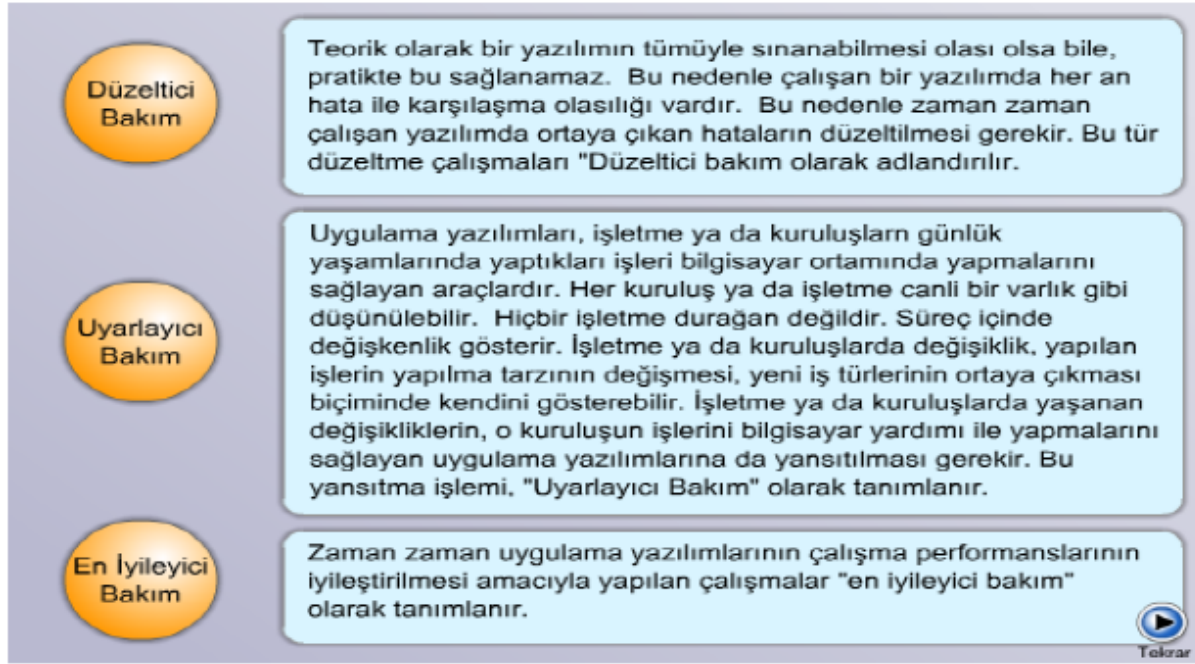
Bu konuyla ilgili pilot bölgede bizzat desteği ben vereceğim bunun yanında sistem canlandırılıp gerçeğe geçirilirse sistem tanımlaması kurulum için bölgelerde bayilik sistemi gibi alt kuruluşlara yetki verilecek eğer profesyonel destek istenirse yol uçak masrafını karşılamak şartıyla bölgeye yetkili gönderilip orada bir organizasyon yapılacaktır.

### 7.4 Yazılım Bakımı



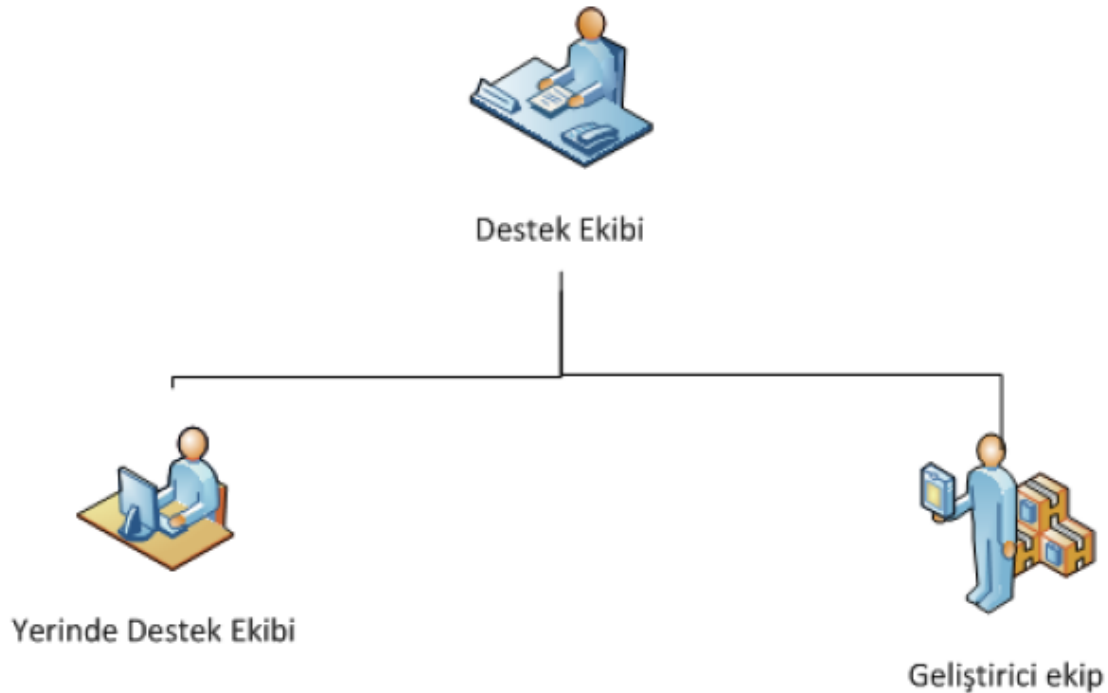
Şekil 7.2 Bakım Aşaması

#### 7.4.1 Tanım



#### 7.4.2 Bakım Süreç Modeli

Aslına bakmak gerekirse bakım süreç modeli yukardaki yapılan işlemlerin tümünün baştan yapılması demek bunları adım adım bir inceleyelim.



## 1. Adım: Sorunu Tanımlama Süreci

İlk önce bakım ne için yapılıyor sorun ne buna bir bakalım.



## 2. Adım: Çözümleme Süreci

Sorun tanımlamadan çıkan karar doğrultusunda problemi kâğıt üzerinde çözelim.



### 3. Adım: Tasarım Süreci

Çözümlenen sistem sonucunda tasarımı güncelleştirmeye geldi sıra.



### 4. Adım: Gerçekleştirim Süreci

Tasarımı yapılan sistemin gerçekleştirmesine sıra geldi.





### 5. Adım: Sistem Sınama Süreci

Artık tekrardan tasarlanan sistemin sınama sürecini tekrar ele almak gerekiyor.



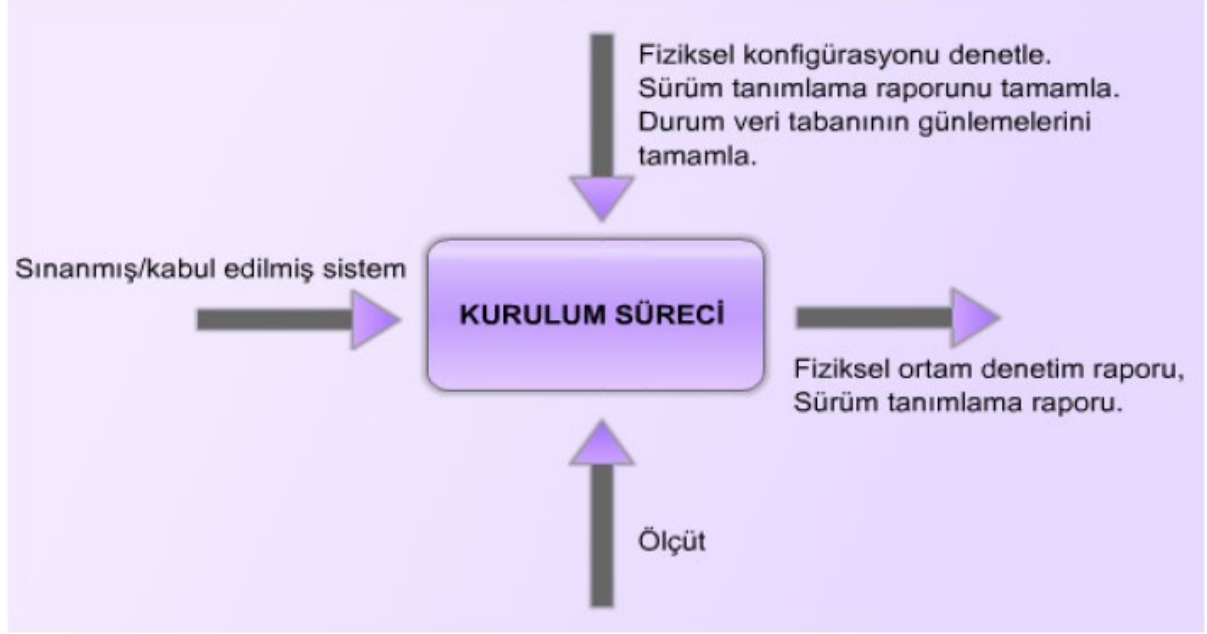
### 6. Adım: Kabul Sınaması Süreci

Kendi içimizde sınavdığımız sistemi birde müşteri karşısında sınavıyoruz.



## 7. Adım: Kurulum Süreci

Kabul sınavasını geçen sistemimiz artık tekrardan kurulum aşamasına geçiyor.



## 8. SONUÇ

Sonuç olarak ilgili kullanıcıların işin kolaylaştıran bir yazılım yarattık.. Sade Ara yüz ve kolay kullanımı sayesinde her kullanıcıya hitap edecek şekilde hazırladık. Bu sayede bu yazılımı kullananlar daha mutlu olacak ve hayatları daha da pratikleşecektir.

Özet olarak;

- Mevcut sistemlerden farkı belirttiğimiz gibi sadece oluşu ve kolay anlaşılabilmesidir.
- En büyük avantajı üzerine basa basa söylediğimiz gibi sade oluşu ve her kesime hitap eden kolay bir kullanımı olması
- Dezavantaj olarak çok büyük sistemlere uygun değildir. Fakat zaman ile otomasyonu daha da detaylandırarak, büyük sistemler içinde aktif bir şekilde kullanılmasını sağlayabiliriz.

## 9. KAYNAKLAR

### İnternet Kaynakları

1. <http://muhammetbaykara.com/2019-2020-bahar-yariyili-dersler/ymh114-yazilim-muhendisliginin-temelleri/>
2. <http://muhammetbaykara.com/2017/04/05/ymh-114-ornek-taslak-proje/>
3. <https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database>
4. <https://www.bmc.com/blogs/software-requirements-specification-how-to-write-srs-with-examples/>
5. <https://www.reqview.com/doc/iso-iec-ieee-29148-srs-example.html>
6. <https://freshcodeit.com/freshcode-post/creating-srs-step-by-step-by-analyzing-requirements>
7. Dr. Ali NİZAM, "Yazılım Proje Yönetimi", ISBN:978-605-4220-74-8 2. basım, Eylül 2015.
8. <http://www.kriptarium.com/pd.html>
9. <https://writingcenter.unc.edu/faculty-resources/classroom-handouts/thesis-analysis/>
10. [https://owl.purdue.edu/owl/general\\_writing/the\\_writing\\_process/thesis\\_statement\\_tips.html](https://owl.purdue.edu/owl/general_writing/the_writing_process/thesis_statement_tips.html)