

# AKILLI EMLAK



AYŞECÜL..

Fırat Üniversitesi

T.C



**FIRAT ÜNİVERSİTESİ**  
**TEKNOLOJİ FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ**  
**YMT218 NESNE TABANLI PROGRAMLAMA**  
**PROJE UYGULAMALARI VE DOKÜMANTASYONU**

**Proje Çalışma Grubu**

**Ayşegül SELEK**

**Danışman: Doç. Dr. İbrahim Türkoğlu**

**Proje Ödevinin Verildiği Tarih: Mayıs 2017**

**Mayıs 2017**

## ÖNSÖZ

Program bir emlak şirketinin simülasyonudur.

Veri tabanı Microsoft Access ile oluşturulmuştur.

Visual Studio 2015 (.NET Framework 4.3) programı kullanılarak C# dilinde gerçekleştirilmiştir.

Programda amaç satılık ve kiralık gayrimenkullerin alıcıların kayıtlarının ve satıcı kayıtlarının yapılarak alıcı ve satıcı arasındaki ikili ilişkilerinde gayrimenkulün satış ve pazarlama safhalarında kullanılmak için hazırlanmıştır.

## İçindekiler..

ÖNSÖZ.....	II
İçindekiler.....	III
1.GİRİŞ.....	- 1 -
1.1 Projenin Amacı .....	- 1 -
1.2 Projenin Kapsamı.....	- 1 -
1.3 Tanımlamalar ve Kısıtlamalar .....	- 1 -
2.Proje Planı .....	- 1 -
2.1 Projenin Plan Kapsamı.....	- 2 -
2.2 Proje Zaman ve İş Plan .....	- 5 -
2.3 Proje Ekip Yapısı .....	- 6 -
2.4 Kullanılan Özel Geliştirme Araçları ve Ortamları.....	- 7 -
2.5 Proje Standartları Yöntem ve Metodolojiler.....	- 8 -
2.5.1 Sprial Modelin Yapısı .....	- 8 -
2.5.2 Sprial Modelin Seçilme Sebepleri .....	- 9 -
2.6 Kalite Sağlama Planı.....	- 9 -
2.6.1 Kalite Güvence .....	- 10 -
2.6.2 Kalite Kontrol .....	- 10 -
2.7 Eğitim Planı .....	- 11 -
2.8 Test Planı .....	- 11 -
2.9 Bakım Planı.....	- 11 -
3. SİSTEM ÇÖZÜMLEME.....	- 12 -
3.1 Proje gereksinimi .....	- 12 -
3.2 Mevcut Sistemin çözümlenmesi .....	- 12 -
3.2.1 Örgüt yapısı.....	- 13 -
3.3 İşlevsel Model.....	- 14 -
3.4 Gereksenen Sistemin Mantıksal Modeli .....	- 15 -
3.4.1 Giriş .....	- 15 -
3.4.2 İşlevsel Model.....	- 15 -
3.4.3 Sistem Senaryoları: .....	- 15 -
3.4.4 Bilgi Sistemleri Ve Nesneler .....	- 16 -
3.4.5 Veri Modeli.....	- 16 -
3.4.6 Veri Sözlüğü .....	- 17 -

3.4.7 Başarım Gerekleri .....	- 18 -
3.5 Ara yüz (Modül) Gerekleri .....	- 18 -
3.5.1 Yazılım Ara yüzü .....	- 18 -
3.5.2 Kullanıcı Ara yüzü .....	- 23 -
3.6 Belgeleme Gerekleri .....	- 24 -
3.6.1 Geliştirme Sürecinin Belgelenmesi .....	- 24 -
3.6.2 Eğitim Belgeleri .....	- 24 -
3.6.3 Kullanıcı El Kitapları .....	- 24 -
4. SiSTEM TASARIMI .....	- 25 -
4.1 Giriş .....	- 25 -
4.1.1 Genel Tasarım Bilgileri .....	- 25 -
4.1.2 Sistem Mimarisi .....	- 25 -
4.1.3Veri Arabirimleri .....	- 26 -
4.1.4 Veri Modeli .....	- 26 -
4.1.5 Testler .....	- 26 -
4.2 Veri Tasarımı .....	- 27 -
4.2.1 Tablo Tanımları .....	- 27 -
4.2.2 Tablo-ilişki şemaları .....	- 27 -
4.3 Süreç Tasarımı .....	- 27 -
4.3.1 Genel Tasarım .....	- 27 -
4.4 Ortak Alt Sistemlerin Tasarımı .....	- 28 -
5.SiSTEM GERÇEKLEŞTİRİMi .....	- 28 -
5.1 Yazılım Geliştirme Ortamları .....	- 28 -
5.2 CASE Araç ve Ortamları .....	- 28 -
5.3 Kodlama Stili .....	- 29 -
5.3.1Kod Biçimlemesi .....	- 29 -
5.4.Yapısal Programlama Yapıları .....	- 29 -
5.5 Program Karmaşıklığı .....	- 30 -
5.5.1 Programın Çizgi Biçimine Dönüştürülmesi .....	- 30 -
5.5.2 McCabe Karmaşıklık Ölçütü Hesaplama .....	- 30 -
5.6 Olağan Dışı Durum Çözümleme .....	- 30 -
5.7 Kod Gözden Geçirme .....	- 31 -
5.7.1 Gözden Geçirme Sürecinin Düzenlenmesi .....	- 31 -
5.7.2 Gözden Geçirme Sırasında Kullanılacak Sorular .....	- 31 -

6. DOĞRULAMA VE GEÇERLEME.....	- 33 -
6.1 Giriş Doğrulama, .....	- 33 -
6.2 Sınama Kavramları .....	- 33 -
6.2.1 Birim Sınama .....	- 33 -
6.2.2 Alt-Sistem Sınama .....	- 34 -
6.2.3 Sistem Sınaması .....	- 34 -
6.2.4 Kabul Sınaması .....	- 34 -
6.3 Doğrulama ve Geçerleme Yaşam Döngüsü .....	- 34 -
6.4 Sınama Yöntemleri .....	- 35 -
6.4.1 Beyaz Kutu Sınaması .....	- 35 -
6.4.2 Temel Yollar Sınaması .....	- 35 -
6.5 Sınama ve Bütünleştirme Stratejileri .....	- 36 -
6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme .....	- 36 -
6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme .....	- 36 -
6.7 Sınama Belirtileri Öncesi .....	- 37 -
6.7.1 Yaşam Döngüsü Boyunca Sınama Etkinlikleri .....	- 38 -
7.BAKIM.....	- 39 -
7.1 Giriş: .....	- 39 -
7.2 Yazılım Bakımı .....	- 39 -
7.3 Tanım .....	- 42 -
7.4. Bakım süreç modeli .....	- 43 -
8.SONUÇ.....	- 44 -
9.KAYNAKLAR.....	- 44 -

## 1.GİRİŞ

### 1.1 Projenin Amacı

Akıllı emlak otomasyonu projemizdeki ana amaç satılık ve kiralık gayrimenkullerin alıcıya satış yapmak için çok işlev düzeyinde sunmak, satıcıya ise gayrimenkulünü en güzel şekilde ve isteğine uygun yollarda satmak için ilan verip kayıt altına almak. Bunları da en iyi şekilde sunmak için küçük çaplı simülasyonumuzdan yardım almaları için Visual Studioda C# dilimizle geliştirdik.

### 1.2 Projenin Kapsamı

Projemizin kapsamı Visual Studio 2015 ara yüzü ile C# dili bünyesinde Windows form özelliğini kullanarak amaca en uygun şekilde tasarlanmıştır.

### 1.3 Tanımlamalar ve Kısıtlamalar

Satılık =satılık menüsü “konut,bina,işyeri,arsa,turistik tesis” .

Kiralık=kiralık menüsü “konut,bina,işyeri,arsa,turistik tesis”.

Müşteri alıcı=alıcı gayrimenkul kayıt etme , silme,resim ekleme .

Müşteri satıcı = satıcı gayrimenkul kayıt etme ve silme,resim ekleme..

Arama=Fiyat aralığına ,Metrekaresine,Emlak tipine,Isıtmaya göre gayrimenkul bulma .

## 2.Proje Planı

Akıllı emlak programımızda ilk olarak kullanıcı giriş emlakçı tarafında girildikten sonra emlak kontrolü formu penceresinden alıcı ve satıcı belirlenip ona göre seçim yapılmalı.Eğer satıcı ise kiralık veya satılık sekmesinden konut,bina,işyeri,arsa,turistik tesis kısmında açılan pencereden gayrimenkule uygun özellikler belirlenip kayıt altına alınarak alıcı geldiği zaman emlak kontrolü formumuzda arama butonuna yönlendirilebilir.Aynı şey alıcı açısından da geçerli fakat dediğimiz gibi alıcıya alternatif olarak arama formu ve müşteri alıcı butonu koyuldu .Arama formundan istediğin özelliklere uygun gayrimenkul aratabilir resmini görebilir sonrası emlakçıyla iletişimine kalmıştır.

## 2.1 Projenin Plan Kapsamı

### Teknik Karmaşıklık Tablosu

**0:** Hiçbir etkisi yok   **1:** Çok az etkisi var   **2:** etkisi var   **3:** ortalama etkisi var   **4:** Önemli etkisi var   **5:** Mutlaka olmalı kaçınılmaz

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



## Maliyet kestirim hesabı

### Maliyet Kestirim Tablosu

Ağırlık Faktörü						
ÖLÇÜM PAREMETRESİ	Sayı	Yalın	Ortalama	Karmaşık	Ağır. Ort.	Toplam
Kullanıcı girdi sayısı	2	3	4	6	4	8
Kullanıcı çıktı sayısı	6	4	5	7	5	30
Kullanıcı sorgulama sayısı	10	3	4	6	4	40
Kütük sayısı	4	7	10	15	7	28
Dışsal arayüz sayısı	1	5	7	10	7	7
Toplam						113

Yukarıdaki tabloda girdi, çıktı ve sorgulama sayılarına göre işlev noktasında ağırlıklı işlev noktasını hesaplarız.

- $KGS = YLN * ORT * KAR = 8$
- $KÇS = YLN * ORT * KAR = 30$
- $KSS = YLN * ORT * KAR = 40$
- $KS = YLN * ORT * KAR = 28$
- $DAYS = YLN * ORT * KAR = 7$
- $AİN = KGS + KÇS + KSS = 113$

Olarak bulunur

### 1.Aşama

#### İşlev Noktası Hesaplama

İN: işlev Nokta Sayısı, AİN: Ana İşlev Nokta Sayısı, TKF: Teknik Karmaşıklık Faktörü

$$İN = AİN \times (0.65 * 0.01 * TKF) \quad (E.1)$$

$$\text{Toplam} \times (0.65 * 0.01 * 53) = (113 * (0,65 * 0,01 * 53)) = 38,92 \quad (E.1)$$

$$\text{Tahmini oluşacak satır sayısı: Satır Sayısı} = İN * 30 \quad (E.2)$$

$$\text{Satır Sayısı} = 49.764 * 30 = 1.167,85 \quad (E.2)$$

### Etkin Maliyet Modeli – COCOMO

İş gücü ( K) $K=a*s^b$  ve Zaman (T) $T=c*K^d$

a,b,c,d: her bir model için farklı katsayılar

S=bin türünden satır sayısı

Tasarladığım sistem boyutu küçük ,deneyimli personel tarafından gerçekleştirilmiş,LAN üzerinde çalışan insan kaynakları yönetim sistemi olduğu için sistemimi Ayırık sistem olarak ele alırsam;

$$\text{İş gücü ( K)} K = 2.4 * 1^{1,05} = 2,4 \text{ iş gücü} \quad (\text{E.3})$$

$$\begin{aligned} \text{Zaman } T &= 2.5 * 2^{0,38} \\ &= 3.2 \text{ ay} \end{aligned} \quad (\text{E.4})$$

## 2. Aşama

### Maliyet Çarpanı

Maliyet çarpanı (C) 15 maliyet etmeninin (Ci) çarpımı sonucu elde edilir.

$$C = C1 * C2 * C3 * ... * C15 \quad (\text{E.5})$$

**Ürün özelliklerinden :** Rely=1,00 , Data=1,00, Cplx=1.00

**Bilgisayar Özellikleri:** Time=1.11, Stor= 1.00 , Vırt=1,00 ,Turn=1.07

**Personel Özellikleri :** Acap= 0.86 , Aexp= 1.00 ,Pcap=1.00 ,Vexp=1.10 ,Lexp=1.07

**Proje Özellikleri:** Modp= 1,10 , Tool= 1,00 ,Sced=1.08

$$C = C1 * C2 * C3 * ... * C15 = 1.428$$

## 3. Aşama :

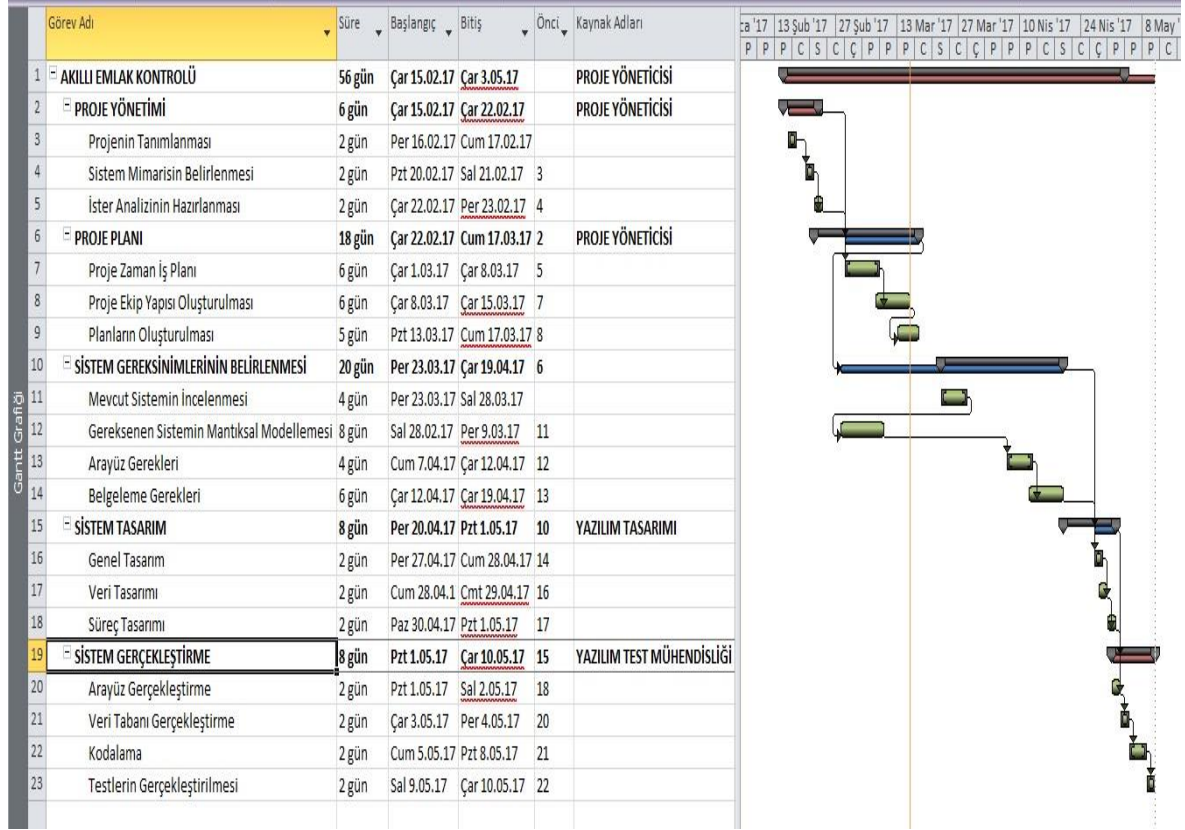
### İlk İş Gücü Değerlerini Düzeltme

Maliyet çarpanı (C),2. Adımda elde edildikten sonra ilk aşamada elde edilen iş gücü değeri (K)ile çarpılarak düzeltilmiş iş gücü Değeri (Kd) elde edilir.

$$Kd = K * C \quad (\text{E.6})$$

$$Kd = 3.4272$$

## 2.2 Proje Zaman ve İş Plan



## 2.3 Proje Ekip Yapısı

### ✓ Proje Yöneticisi

Proje yöneticisi yazılım ekibini bir arada tutan ve zaman çizelgelerine uyulması için gerekli motivasyonu sağlayan kişidir. Ayrıca yönetim ile proje ekibi arasındaki bilgi alışverişinin de sağlar. Bütçe konularında düzenlemeler ve maliyet analizleri konusunda yönetim kuruluna bilgi ve tavsiye verir. Yazılacak modüllerin ve ara yüzlerin zorluk derecelerine göre zamanlarını tayin eder ve proje planı içinde yayınlar. Riskleri belgeleyerek çözümler için onaya sunar.

### ✓ Kalite Uzmanı

İhtiyaçların ve geliştirilen çözümün doğru belirlenip belirlenmediğini, yazılımın belirli standartlarda olup olmadığını denetleyen kişidir. Yazılım tasarımı ve/veya yazılım testi konularında bilgi sahibidir. Genel kalite yönetim sistemi standartlarını, uluslararası yazılım mühendisliği standartlarını ya da süreç olgunluk modellerini bilir. Geliştirilen yazılımın bunlara uygun olarak yürümesini sağlar.

### ✓ Programcı

Yapılan analizlere göre belirlenen teknolojiyi, platformu kullanarak yazılımı kodlayan kişidir.

### ✓ Yazılım Destek Elemanı

Yazılım destek elemanı, satılmış olan yazılım ürünlerini müşteri bilgisayarlarına yüklemek, veri tabanını kurmak, satış yapıldıktan sonra ücretli olarak danışmanlık ve destek hizmetlerini müşteri talebi doğrultusunda yerine getirme görevini üstlenmiştir.

### ✓ Donanım Ekip Lideri

Kullanılacak donanımları en düşük maliyetle ve en verimli şekilde tespit etmeye çalışır. Donanım mühendislerini ve destek elemanlarını kontrol eder.

### ✓ Donanım Mühendisi

Bilgisayar donanım mühendisleri araştırma, geliştirme tasarım ve çeşitli bilgisayar donanımlarını test eder. Bazı güncellemeler yaparak mevcut bilgisayar donanımını yazılımları daha iyi çalıştırmaya yönlendirir.

### ✓ Donanım Destek Elemanı

Bilgisayar donanımlarında çıkan sorunları tespit edip engellemeye, düzeltmeye çalışır.

#### ✓ **Bilgisayar Ağ Uzmanı**

Network topolojisiyle ilgilenir. Uzak yerlerdeki bilgisayarları birbirlerine bağlar (WAN ağı kurar). Ağ en ucuz şekilde en uzak mesafeye ve en hızlı çalışacak şekilde bağlamaya çalışır.

#### ✓ **Sistem Çözümleyici**

Bilgi işlem sistemlerini kuran ve yeni bilgi toplayan, sistemlerin kurulmaları ve çalışmaları için gerekli yöntemleri tanımlayan, kurulumlarını yapan, denetleyen ve gelişmeleri için önerilerde bulunan nitelikli kişi.

#### ✓ **Sistem Tasarımcı**

Sistem çözümleyicinin tanımladığı gereksinimleri mantıksal, ekonomik ve pratik sistem tasarımlarına dönüştürerek ilgili programların yazılabilmesi için gerekli ayrıntılı spesifikasyonları hazırlayan kişidir

#### ✓ **Sistem Yöneticisi**

Sistem yöneticisi, projenin ihtiyaçlarını analiz ederek bilgisayar sistemlerini tasarlama, kurma, destekleme, geliştirme, sürekliliğini ve güvenliğini sağlama işini yapar.

#### ✓ **Veri Tabanı Yöneticisi**

Veri tabanı sistemlerinin kurulması, konfigürasyonun yapılması, tasarlanması,

### **2.4 Kullanılan Özel Geliştirme Araçları ve Ortamları**

- Visual Studio 2015
- Microsoft Access
- Argo UML
- Project Pro

## 2.5 Proje Standartları Yöntem ve Metodolojiler

Yazılım geliştirme sürecinin yapısını ve adımlarının uygulanış biçimini, seçilen yazılım geliştirme modeli belirlemektedir. Dikkat edilmesi gereken nokta eldeki ürüne ve sürece uygun modelin seçilebilmesidir. Bu projede, kişisel bilgi yönetim yazılımı olarak bir ajanda uygulaması oluşturmak için yazılım geliştirme modellerinden biri olan spiral (sarmal) model kullanılır.



Helezonik model, bir anlamda çağlayan modelinin ve prototipleme yaklaşımının birleştirilmiş şekli olarak düşünülebilir. Söz konusu yaklaşımlarda yeterince vurgulanmayan risk çözümleme olgusu, Helezonik modelde ön plana çıkarılmıştır. Helezonik model temelde sistemi kullanıcı açısından anlamlı parçalara ya da ara ürünlere bölme ve her bir ara ürün için, Planlama, Risk Çözümleme, Üretim ve Kullanıcı Değerlendirmesi adımlarını gerçekleştirme adımlarına dayanır.

### 2.5.1 Sprial Modelin Yapısı

#### Planlama:

- Amaç belirlenir,
- Alternatifler belirlenir,
- Kısıtlar belirlenir.

**Risk Çözümleme:**

- Alternatifler değerlendirilir,
- Risk analizi yapılır.

**Üretim:**

- Geliştirme yapılır,
- Bir sonraki ürün belirlenir.

**Kullanıcı Değerlendirmesi:**

- Bir sonraki aşama planlandırılır
- Kullanıcı değerlendirme alınır

**2.5.2 Spiral Modelin Seçilme Sebepleri**

Spiral modelin bu projede seçilme nedenleri ve diğer modellere göre faydaları;

1. Kullanıcı Katkısı Üretim süreci boyunca ara ürün üretme ve üretilen ara ürünün kullanıcı tarafından sınanması temeline dayanır. Yazılımı kullanacak personelin sürece erken katılması ileride oluşabilecek istenmeyen durumları engeller.
2. Yönetici Bakışı Gerek proje sahibi, gerekse yüklenici tarafındaki yöneticiler, çalışan yazılımlarla proje boyunca karşılaştıkları için daha kolay izleme ve hak ediş planlaması yapılır.
3. Yazılım Geliştirici (Mühendis) Bakışı Yazılımın kodlanması ve sınanması daha erken başlar.

**2.6 Kalite Sağlama Planı**

Buna rağmen, yazılım kalitesi basit bir yolla tanımlanması mümkün olmayan karmaşık bir kavramdır. “Klasik olarak, kalite kavramı, üretilen ürünün belirtilmelerini karşılaması gerektiğini ortaya koyar (Crosby, 1979).” Kalite sağlama üzerine geliştirilmiş belirtilmeler gerçek dünyadaki çoğu ürün için uygulanabilse de yazılım için istenilen seviyeye ulaşamamış, tam olarak kaliteyi ölçmekte yararlı olamamışlardır. Bu projede yazılım kalite yönetimi üç temel davranış ile yapılandırılacaktır:

**Kalite Güvence:** Yüksek kaliteye sahip yazılıma götürecek kurumsal yordam ve standartlar çatısının ortaya koyması.

**Kalite Planlama:** Bu çatıdan uygun standart ve yordamların seçimi ve bunların projeye uyarlanması.

**Kalite Kontrol:** proje kalite standart ve yordamlarının yazılım geliştirme takımı tarafından takip edildiğini garanti eden süreçlerin tanımlanması ve belgelenmesi.

### 2.6.1 Kalite Güvence

Kalite güvence (KG) etkinlikleri yazılım kalitesini başarmak için çatı tanımlar. KG süreci, yazılım geliştirme süreci veya yazılım ürününe uygulanacak standartlar seçmeyi veya tanımlamayı içerir. Bu standartlar geliştirme süresince uygulanacak yordam veya süreçlerin içine gömülmüş olabilir.

**1. Ürün Standartları:** Bunlar geliştirilecek projeye uygulanacak standartlardır. Üretililecek gereksinim belgesinin yapısı gibi belge standartları, bir nesne sınıf tanımlaması için standart yorum başlığı gibi belgeleme standartları ve bir programlama dilinin nasıl kullanılabileceği gibi kodlama standartları bunlara dahildir.

**2. Süreç Standartları:** Bunlar yazılım geliştirme süresince takip edilecek süreçleri tanımlayan standartlardır. Belirtim, tasarım ve doğrulama süreçlerinin tanımları ve bu süreçler süresince oluşturulması gereken belgelerin tanımları bunlara dahildir. 2.8.2 Kalite Planlama Kalite planlama yazılım sürecinin erken bir evresinde başlanacaktır. Kalite planı istenen ürün kalitelerini ortaya koyar. Bunlara nasıl değer biçileceğini de tanımlamalıdır. Bu yüzden yüksek kaliteli yazılımın gerçekte ne anlama geldiğini tanımlar. Böyle bir tanım olmazsa, farklı mühendisler zıt yönlerde çalışabilirler böylece farklı ürün özellikleri en iyilenir. Kalite planlama sürecinin sonucu bir proje kalite planıdır. Projede şu taslak üzerine gidilecektir:

1. Ürün başlangıcı Ürünün, sunulacak pazarın ve ürün için kalite beklentilerinin tanımları
2. Ürün planları Önemli sürüm tarihleri ve ürün sorumlulukları yanında dağıtım ve ürün bakım planları
3. Süreç tanımlamaları Ürün geliştirme ve yönetimi için kullanılacak geliştirme ve bakım süreçleri
4. Kalite hedefleri Önemli ürün kalite özelliklerinin de tanımlandığı ürün kalite hedef ve planlar
5. Riskler ve risk yönetimi Ürün kalitesini etkileyebilecek anahtar riskler ve bu riskleri karşılayan eylemler. Kalite planları yazılırken bunların olabildiğince kısa olmasına dikkat edilecektir.

### 2.6.2 Kalite Kontrol

Kalite kontrol, kalite güvence yordam ve standartlarına uyulmasını garanti etmek için yazılım geliştirme sürecinin gözden geçirilmesini gerektirir. Kalite kontrol sürecinin yazılım geliştirme sırasında kullanılması gereken kendi yordam ve rapor kümesi vardır. Bu yordamlar yazılımı geliştiren mühendisler tarafından düzgün ve kolaylıkla anlaşılabilir olmalıdır. Kalite kontrole iki çeşit tamamlayıcı yaklaşım vardır:



1. Yazılımı geliřtirmek için kullanılan belgeleme ve süreçlerin kalite incelemesi bir grup insan tarafından yapılacaktır. Bu kişiler proje standartlarının izlenmesinin ve yazılım ile belgelerin standartlara uygunluğunun kontrolünden sorumlu olacaktır. Standartlardan sapmalar kayıtlanmalı ve proje yönetiminin dikkatine sunulmalıdır.

2. Üretilen yazılım ve standartların bir program tarafından yapılır ve geliştirme projesine uygulanan standartlarla karşılaştırılırsa buna otomatikleştirilmiş yazılım değerlendirme adı verilecektir. Otomatikleştirilmiş değerlendirme bazı yazılım özelliklerinin nicel ölçümlerini gerektirir. 2.9 Eğitim Planı Uygulama basit bir ara yüze sahip olduđu için ve girdi çıktı birimlerinin sabitliğı bir eğitim verilmesine gerek duyulmamıştır. Program indirilirken kullanımını anlatan adımları sırasıyla gösterilecektir.

## 2.7 Eğitim Planı

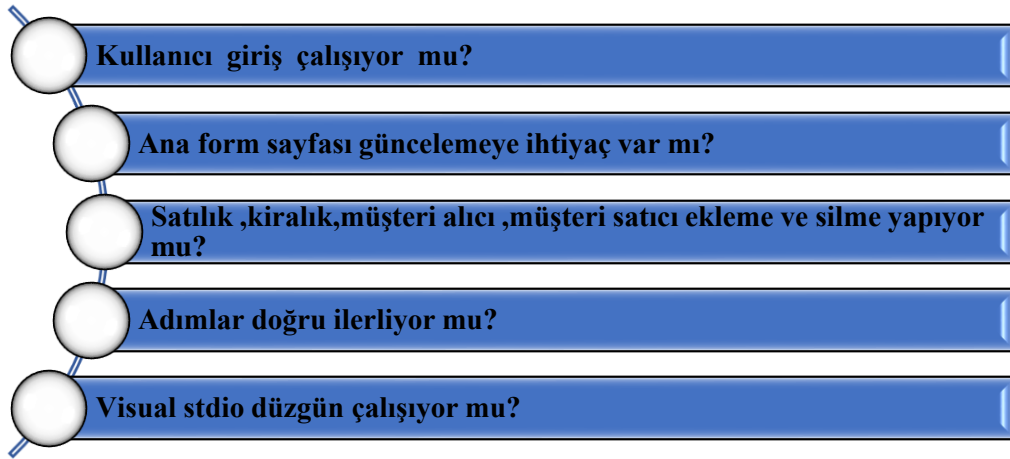
Uygulama basit bir ara yüze sahip olduđu için ve girdi çıktı birimlerinin sabitliğı bir eğitim verilmesine gerek duyulmamıştır. Program indirilirken kullanımını anlatan adımları sırasıyla gösterilecektir.

## 2.8 Test Planı



## 2.9 Bakım Planı

. Sistem, ilk 1 sene her ay bakıma alınacaktır. 1 seneden sonra ciddi sorun çıkarabilecek hatalar düzeltileneğı için 3 ayda 1 bakım yapılacaktır. Bakım yapılırken řu şartlar aranacaktır:



### 3. SİSTEM ÇÖZÜMLEME

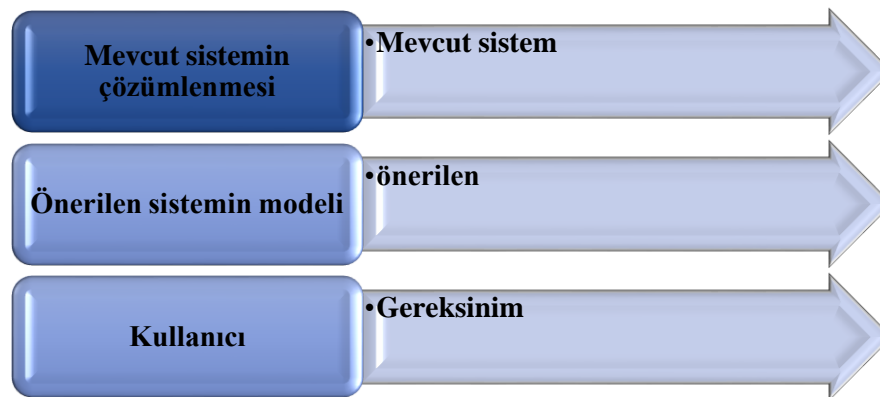
#### Giriş

Sistem çözümleme çalışması, üretim sürecinin başlangıcıdır. Bu aşamada temel olarak mevcut sistemin nasıl çalıştığı araştırılır. Mevcut sistemin incelenmesi sırasında temel hedef gereksinimlerin saptanmasıdır

#### 3.1 Proje gereksinimi

Projenin gereksinim amacı

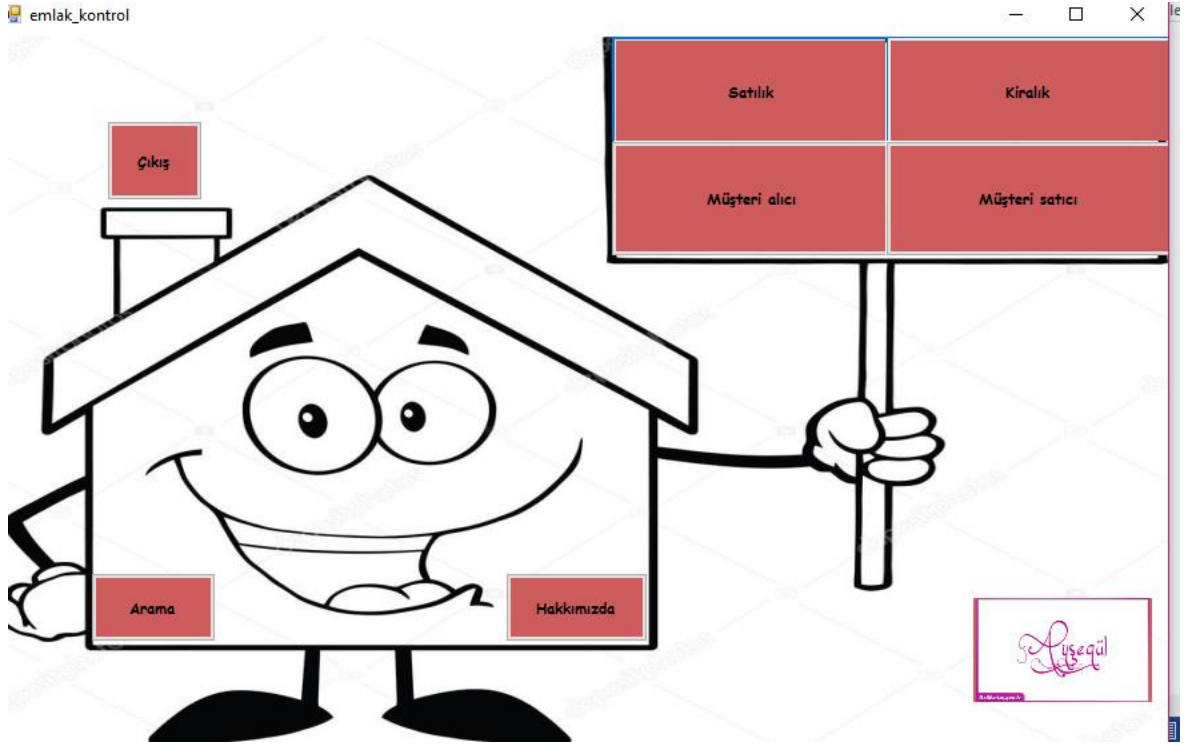
- Oluşturulacak sistemde nelerin olabileceğini belirlemek.
- Daha önce kullanılan sisteme kıyasla nelere ihtiyaç olabileceğini belirlemek.
- Mevcut sistemin varsa bunun yazılımın araştırmasını yapmak.
- Yeni oluşturulacak sistemde ne tür veri ve işlemler gerekeceğinin belirlemek.
- Kullanıcıların anlayacağı şekilde çeşitli işlemlerin belirtilmesi.
- Gereksinimlerin donanımsal olmadan belirlenmesi.
- **Sistem Çözümlemesi Çalışması**



#### 3.2 Mevcut Sistemin çözümlenmesi

Projenin gereksinimlerin araştırılması, tanımlanması, ortaya çıkarılması ve düzgün bir şekilde, terimsel olarak açıklanması bu bölümde yapılacaktır

### 3.2.1 Örgüt yapısı



### 3.3İşlevsel Model

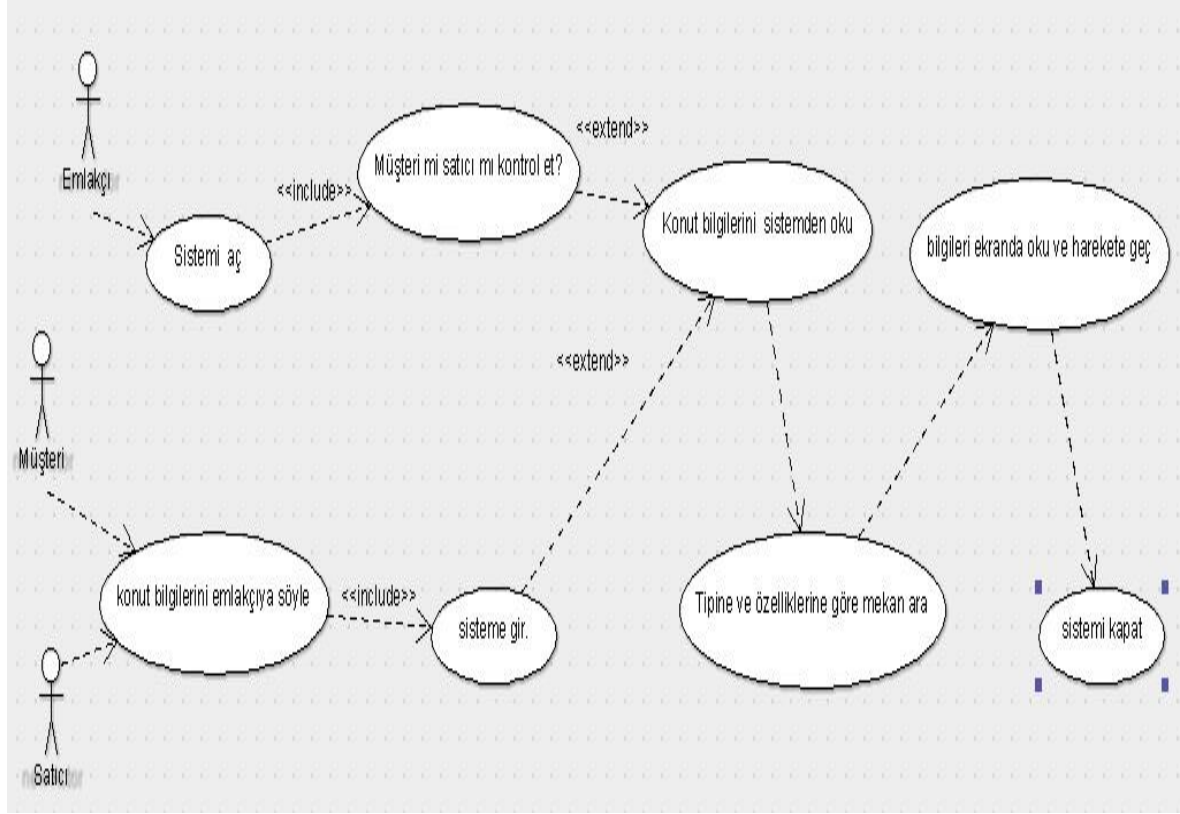


### 3.4 Gereksenen Sistemin Mantıksal Modeli

#### 3.4.1 Giriş

Bu bölümde önerilen sistemin işlevsel yapısı, veri yapısı ve kullanıcı ara yüzünde çözümleme yapılır. Bu model daha çok bilgi sistemini geliştirecek teknik personele yöneliktir. Mantıksal model olarak da tanımlanır.

#### 3.4.2 işlevsel Model



#### 3.4.3 Sistem Senaryoları:

##### Senaryo 1: Kodun derlenmesi

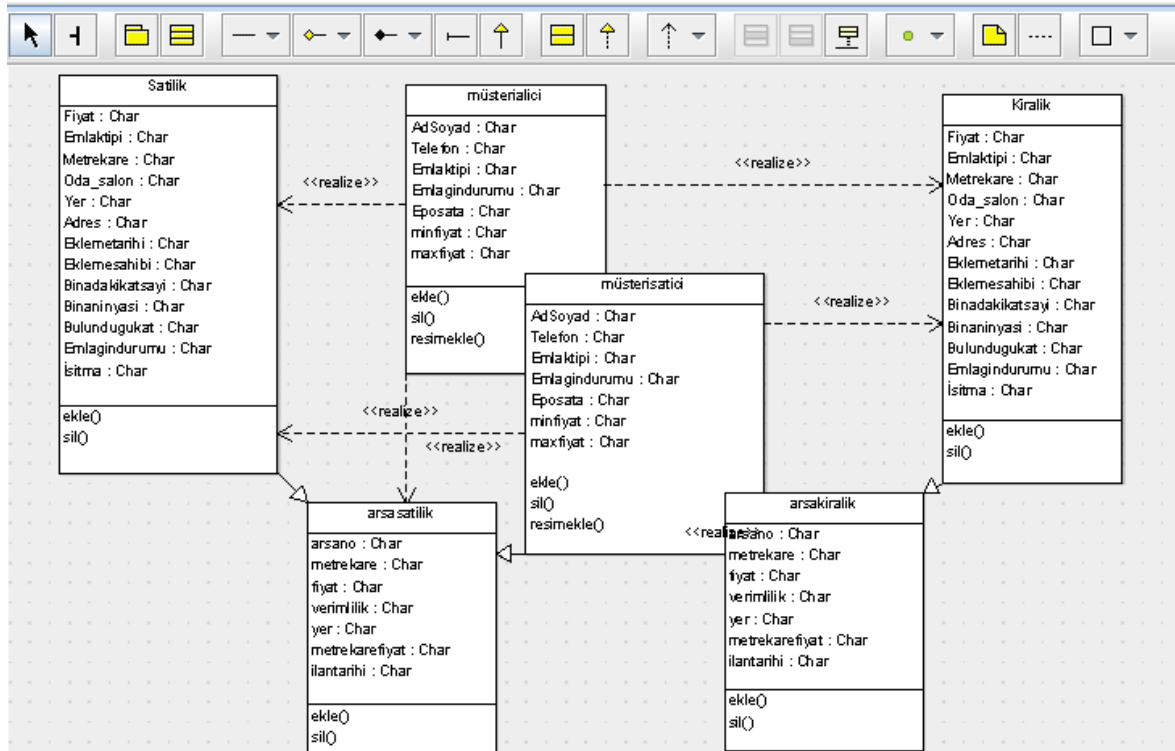
**Aktör:** Emlakçı

**Amaç:** Satıcıyı ,alıcıyı ona göre devam et

Ana Senaryo:

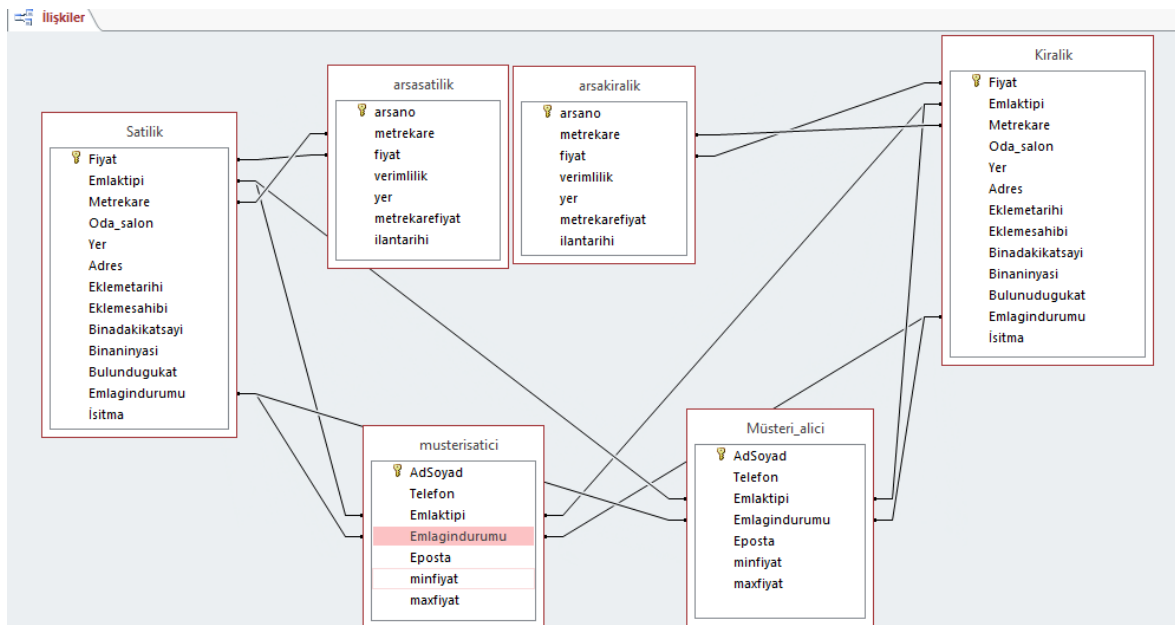
1. Kullanıcı girişini yap .
2. Satıcı veya alıcı olduğunu belirle.
3. Bilgileri ekrana bas.
3. Lojik ifade kod adımlarına göre ilerler.
4. Cevap olarak formdan ifadeyi ekrandan okur .

### 3.4.4 Bilgi Sistemleri Ve Nesneler



### Sınıf Diyagramı

### 3.4.5 Veri Modeli



### 3.4.6 Veri Sözlüğü

arsakiralik		
Alan Adı	Veri Türü	
arsano	Otomatik Sayı	Kullanıcı tablosunun birincil anahtarıdır
metrekare	Kısa Metin	arsanın metrekaresi tutulacak
fiyat	Kısa Metin	arsanın fiyatı tutulacak
verimlilik	Kısa Metin	arsanın özelliği tutulacak
yer	Kısa Metin	arsanın mekanı tutulacak
metrekarefiyat	Kısa Metin	arsanın metrekare başına düşen fiyatı
ilantarihi	Kısa Metin	yayınlanan tarihi

arsasatilik		
Alan Adı	Veri Türü	
arsano	Otomatik Sayı	arsasatilik tablosunun birincil anahtarıdır
metrekare	Kısa Metin	arsanın metrekaresi tutulacak
fiyat	Kısa Metin	arsanın fiyatı tutulacak
verimlilik	Kısa Metin	arsanın özelliği tutulacak
yer	Kısa Metin	arsanın mekanı tutulacak
metrekarefiyat	Kısa Metin	arsanın metrekare başına düşen fiyatı
ilantarihi	Kısa Metin	yayınlanan tarihi

Kiralik		
Alan Adı	Veri Türü	
Fiyat	Otomatik Sayı	kiralık tablosunun birincil anahtarı.
Emlaktipi	Kısa Metin	kiralıkların emlağın türüne göre seçme.
Metrekare	Kısa Metin	kiralıkların metrekaresi tutulacak
Oda_salon	Kısa Metin	kiralığın oda artı salon özelliği tutulacak
Yer	Kısa Metin	kiralığın mekanı il türünden tutulacak
Adres	Kısa Metin	kiralığın ilçe/merkezi tutulacak
Eklemetarihi	Kısa Metin	kiralığın ekleme tarihi
Eklemesahibi	Kısa Metin	kiralığın sahibi
Binadakikatsayi	Kısa Metin	kat sayısı tutulacak
Binaninyasi	Kısa Metin	binanın yası
Bulunudugukat	Kısa Metin	binanın bulunduğu kat
Emlagindurumu	Kısa Metin	emlağın özellikleri
Isitma	Kısa Metin	ısıtma türünden emlaklar

Satilik			Açıklama (İsteğe Bağlı)
Alan Adı	Veri Türü		
Fiyat	Otomatik Sayı	Satilik tablosunun birincil anahtarı	
Emlaktipi	Kısa Metin	satılıkların emlağın türüne göre seçme.	
Metrekare	Kısa Metin	satılıkların metrekaresi tutulacak	
Oda_salon	Kısa Metin	satılıkların oda artı salon özelliği tutulacak	
Yer	Kısa Metin	satılıkların mekanı il türünden tutulacak	
Adres	Kısa Metin	satılıkların ilçe/merkezi tutulacak	
Eklemetarihi	Kısa Metin	satılıkların ekleme tarihi	
Eklemesahibi	Kısa Metin	satılıkların sahibi	
Binadakikatsayi	Kısa Metin	kat sayısı tutulacak	
Binaninyasi	Kısa Metin	binanın yası	
Bulundugukat	Kısa Metin	binanın bulunduğu kat	
Emlagindurumu	Kısa Metin	emlağın özellikleri	
Isitma	Kısa Metin	ısıtma türünden emlaklar	

musterisatici		
Alan Adı	Veri Türü	Açıklama (İsteğe Bağlı)
AdSoyad	Kısa Metin	müşteri satıcının birincil anahtarı
Telefon	Kısa Metin	müşterinin telefon bilgileri tutulacak
Emlaktipi	Kısa Metin	satacağı emlak tipi tutulacak
Emlagindurumu	Kısa Metin	satacağı emalğin durumu tutulacak
Eposta	Kısa Metin	satıcının epostası
minfiyat	Kısa Metin	satılığın min fiyatı
maxfiyat	Kısa Metin	satılığın max fiyatı

Müşteri_alici		
Alan Adı	Veri Türü	Açıklama (İsteğe Bağlı)
AdSoyad	Kısa Metin	müşteri alıcının birincil anahtarı
Telefon	Kısa Metin	müşterinin telefon bilgileri tutulacak
Emlaktipi	Kısa Metin	alacağı emlak tipi tutulacak
Emlagindurumu	Kısa Metin	alınan emalğin durumu tutulacak
Eposta	Kısa Metin	alıcının epostası
minfiyat	Kısa Metin	alıcının min fiyatı
maxfiyat	Kısa Metin	alıcının max fiyatı

### 3.4.7 Başarım Gerekleri

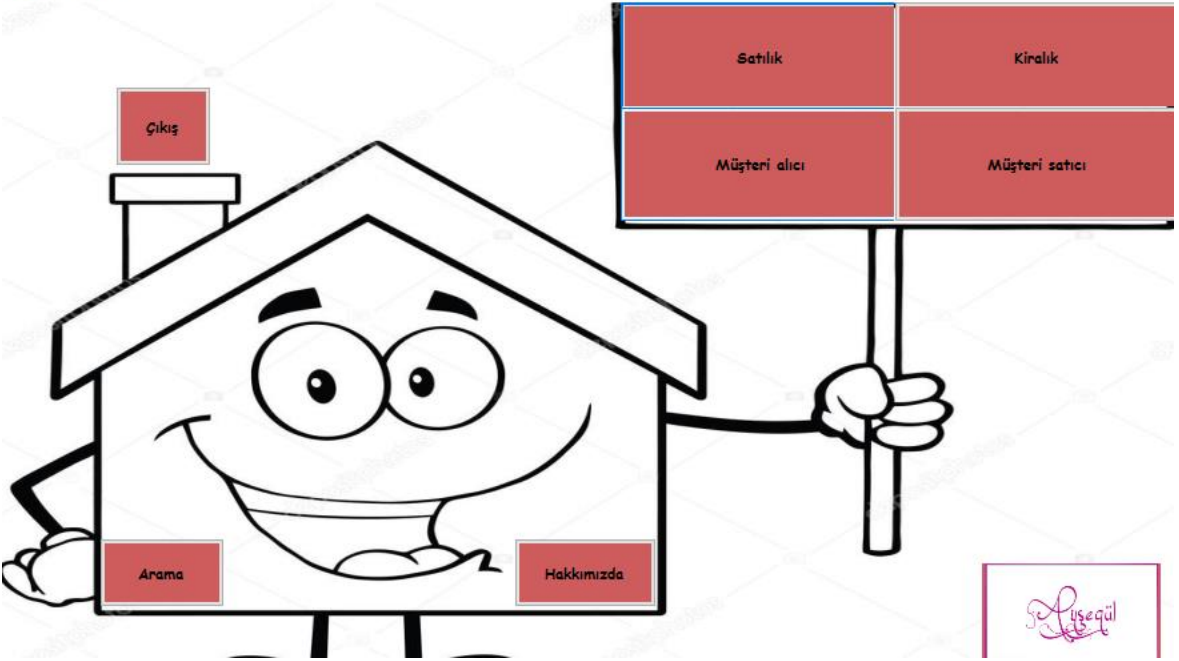
- Sistem ara yüzü karmaşık olmamalı.
- Anlaşılabilir olmalı
- Kullanışlı olmalı.
- Kontrollü olmalı.
- Kısa adımlar içermeli.
- Hızlı çalışabilmeli
- Gerektiğinde sistem sorunu durumunda gerekli ekiplerin bulunması.
- Sistem güncellemesi yapılmalı.
- Sistem kaynaklarını en verimli şekilde kullanmak gibi durumlardır

### 3.5 Ara yüz (Modül) Gerekleri

#### 3.5.1 Yazılım Ara yüzü

Sisteme en uygun yazılım ara yüzü tasarlanarak kullanım kolaylığının sağlanması





satılık\_menüsü



kiralık\_menüsü



Müşteri alıcı

Ad Soyad:



Telefon:

Emlak Tipi:

Emlağın Durumu:

Eposta:

Fiyat Aralığı:  -

 Kaydet  Sil

	AdSoyad	Telefon	Emlaktipi
*			

foto seç Yaz Kaydet

müşteri\_satıcı

Ad Soyad:



Telefon:

Emlak Tipi:

Emlağın Durumu:

Eposta:

Fiyat Aralığı:  -

 Kaydet  Sil

	AdSoyad	Telefon	Emlaktipi
*			

Foto seç Yaz Kaydet

Satılık

Fiyat:  TL

Emlak Tipi:

Metrekare:  m²

Oda+Salon:

Yer(İl/İlçe):

Adres:

Ekleme Tarihi: 3 Mayıs 2017 Çarşamba

Emlak Sahibi:



Binadaki Kat Sayısı:

Binanın Yaşı:

Bulunduğu Kat:

Emlağın Durumu:

Isıtma:

 Kaydet  Sil

	Fiyat	Emlaktipi	Metrekare	Oda_salon
*				

Kiralık

Fiyat:  TL Binadaki Kat Sayısı:

Emlak Tipi:  Binaın Yaşı:

Metrekare:  m<sup>2</sup> Bulunduğu Kat:



Oda+Salon:  Emlağın Durumu:

Yer(İl/İlçe):  Isıtma:

Adres:

Ekleme Tarihi: 3 Mayıs 2017 Çarşamba

Emlak Sahibi:

 Kaydet  Sil

Fiyat	Emlak tipi	Metrekare	Oda_salon
*			

arsa

Arsa No

Metrekare



Fiyat

Verimlilik

Yer

m<sup>2</sup> fiyat

ilan tarihi 3 Mayıs 2017 Çarşamba

 kaydet  sil

arsano	metrekare	fiyat	verimlilik	yer
*				

arsakiralik

Arza No

Metrekare

Fiyat

Verimlilik

Yer

m<sup>2</sup> fiyat

ilan tarihi 3 Mayıs 2017 Çarşamba

	arsano	metrekare	fiyat	verimlilik	yer	met
*						

kaydet sil

Arama

☐ Fiyat aralığı ☐ Metrekare ☐ Emlak tipi ☐ Isıtma

satılığa göre ara

kiralığa göre ara

### 3.5.2 Kullanıcı Ara yüzü

Projenin merkez sistemi şekilsel olarak aşağıdaki şekilde olacaktır. Kullanıcı ara yüzleri tasarlanırken şu noktalara dikkat edilmelidir;

- Pencere ve paneller olabildiğince sade olmalı ve anlaşılır bir dille tasarlanmalıdır.
- Pencereye sığmayan nesneleri kaydırma çubukları ile denetlenmelidir.
- Menüler sol kısımda olmalıdır.

- Sayfalar arası geçişler kullanıcıları sıkmamalı ve her sayfadan geriye dönmelidir.
- Kritik noktalarda kullanıcılardan onay istenmelidir.
- Hata mesajları kullanıcıyı rahatsız etmeyecek şekilde sunulur.
- Benzer görevli düğmeler yan yana veya bir grupta hizasında verilmelidir.

### **3.6 Belgeleme Gerekları**

#### **3.6.1 Geliştirme Sürecinin Belgelemesi**

Belgeleme Microsoft Word ile yapılmaktadır. Bu rapor projenin tüm ayrıntılarını içermektedir. Belge içeriğı aşağıda listelenen 8 ana konudan oluşmaktadır.

- o Giriş
- o Proje Planı
- o Sistem Çözümleme
- o Sistem Tasarımı
- o Sistem Gerçekleştirimi
- o Doğrulama ve Geçerleme
- o Bakım o Sonuç

#### **3.6.2 Eğitim Belgeleri**

Çalışılacak olan robotun eksiksiz ve kullanıma uygun olabilmesi için kullanıcının sistemin bir parçası olabilmesi için eğitim verilmesi gerekmektedir. Bu eğitimin içeriğı; sistemin işleyişı aracın nasıl çalıştığı, kullanıcının sistemdeki yeri, kullanacağı program hakkında bilgi verilmesi olacaktır.

#### **3.6.3 Kullanıcı El Kitapları**

Kullanıcı el kitabında sistemin tanıtımı, amacı ve kullanacağı uygulama hakkında bilgiler olacaktır

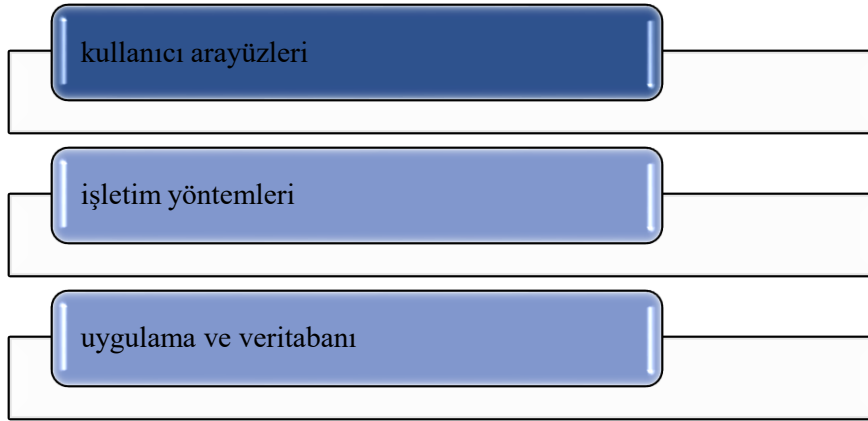
## 4. SİSTEM TASARIMI

### 4.1 Giriş

Tasarım bir konuda yapılması gereken faaliyetleri belirli planlar çerçevesinde uygulamaktır. Yeni bir düzen içerisindeki bilgilerin, organize edilmesi, planlanması ve etkili olarak uygulanması faaliyetleridir

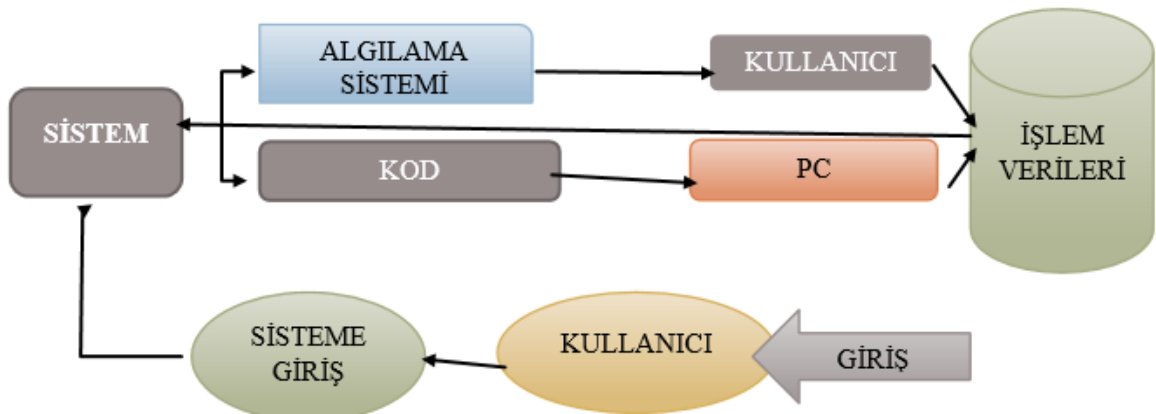
#### 4.1.1 Genel Tasarım Bilgileri

Sistem Model ViEW CONTROLLER mimarisi ele alınarak geliştirilmiştir. On katmanda kullanıcı ara yüzleri bulunurken orta katmanda işlem yönetimleri son katmanda ise uygulamalar ve veri tabanı sistemleri yer almaktadır. MVC yapısının seçilme nedeni hızlı geliştirme, daha iyi performans, sınaıa kolaylığı ve kolay yönetim olanakları sağladığı için tercih edilmiştir.



#### 4.1.2 Sistem Mimarisi

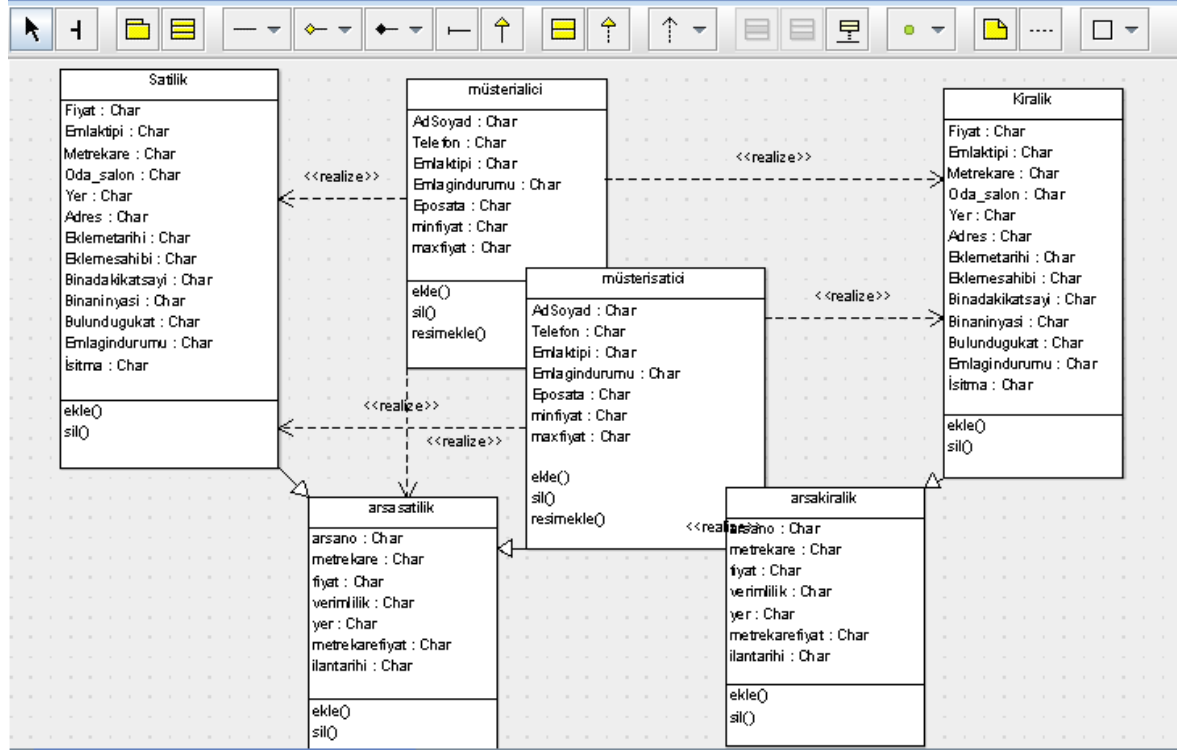
Sistem Model Visual Studio mimarisi ele alınarak geliştirilmiştir. Kodun ve donanımsal sistemin nasıl çalıştığını kontrol etmek ve test etmek için bu sistem ortamı daha sağlıklı bulduk.



### 4.1.3 Veri Arabirimleri

Kullanıcı tarafından indirgenecek değeri sadeleştirmek için kod programını açıp pc tarafından derlendikten sonra od adımlarına göre ekrana sonucun verilmesi gerekir bu mantıkla tekrar güncellemek istersek tekrar derleme yapabiliriz farklı değerlerde .

### 4.1.4 Veri Modeli



Nesne diyagramı

### 4.1.5 Testler

Veri giriş çıkışına bağlı performans kayıtları tutulacak ve ayda bir kontrol edilecektir. Buna bağlı olarak eğer varsa sistemin aksaklığının kontrolü test edilmiş olacaktır. İşlemlerin düzenli bir şekilde aksamadan işlenmesi yapılan uygulama için büyük önem taşımaktadır.

### Yazılım Testi

Yazılım kodlama aşamasında programcı tarafından oluşabilecek hataları gidermek amacıyla acımasız bir şekilde yapılır.

### Yeterlilik Testi

Yazılımın istenilen şekilde yapılıp yapılmadığını kontrol etmek amacıyla yapılır. Yani yazılım isterleri tam olarak karşılıyor mu sorusuna cevap olarak yapılır.

### Sistem Testi



Yoğun veri akışı altında komple yükleme(load) testleri, normal olmayan koşullarda komple sistemin nasıl davranacağını görmek amacıyla germe(stres) testleri, istemli bir şekilde sistemi çökerterek sistemin nasıl davranacağını tespit etmek amacıyla geri kazanım(recovery) testleri, yazılımın geliştirilmesinde birimde yapay verilerle fabrika kabul testi, sistemin kullanılacağı yerde asıl verilerle kullanım hattı testleri, ve bundan sonra deneme testleri yapılır.

## 4.2 Veri Tasarımı

### 4.2.1 Tablo Tanımları

Satılık: Satılık gayrimenkul özellikleri tutulur.

Kiralık: Kiralık gayrimenkul özellikleri tutulur.

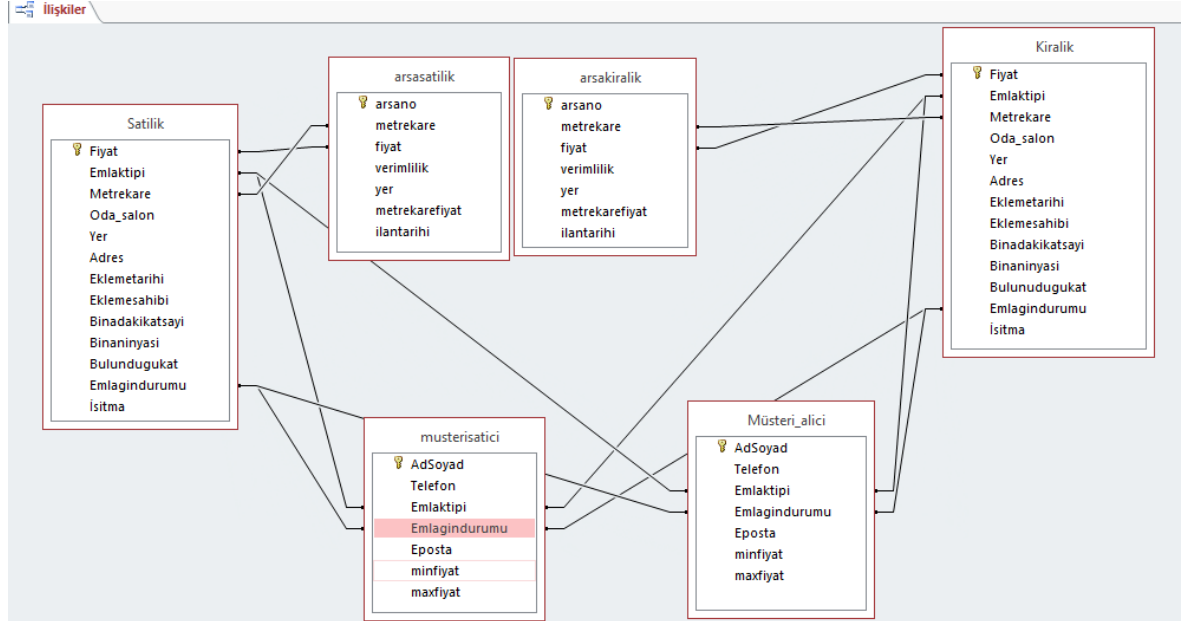
Müşteri Alıcı:Gelen müşterinin isteğe göre satılık ve kiralık gayrimenkul kaydı .

Müşteri satıcı: müşterinin isteğe göre satacağı ve kiralayacağı gayrimenkul kaydı .

Arsa Kiralık: Kiralık arsa kaydı

Arsa satılık:Satılım arsa kaydı

### 4.2.2 Tablo-ilişki şemaları



Tablo ilişki şeması

## 4.3 Süreç Tasarımı

### 4.3.1 Genel Tasarım

Yazılım geliştirme süreçleri sürekli yaşayan ve iyileştirilen süreçler olmak zorundadır.

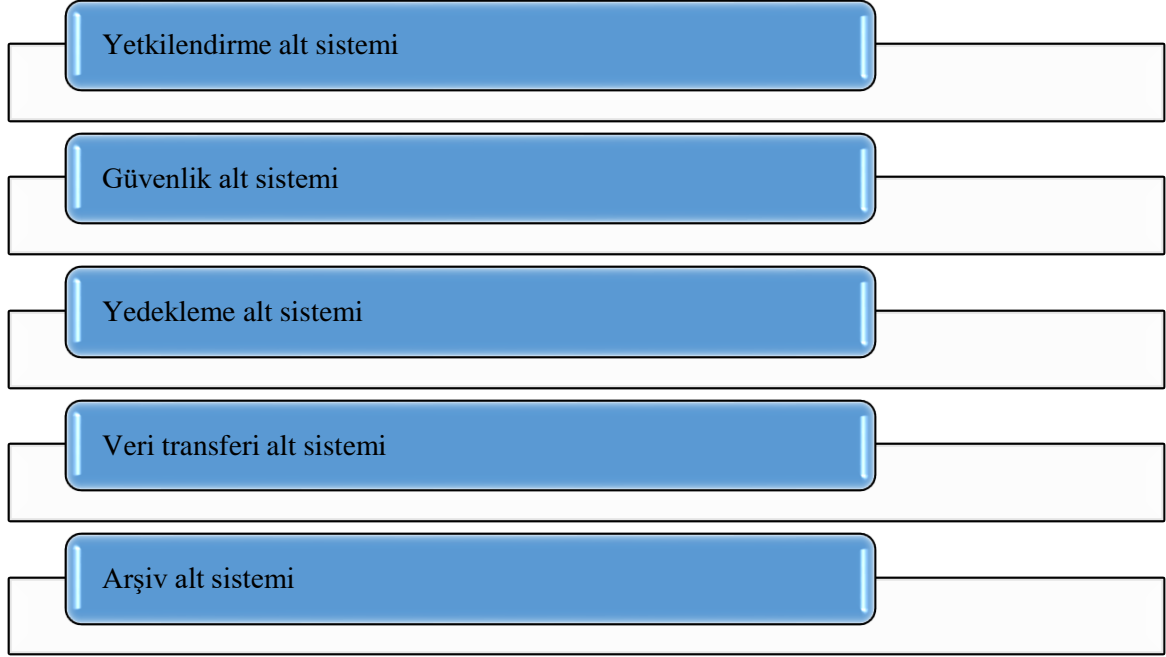
Dolayısıyla bir süreç mekanizması kurulması zorunludur. Süreç yönetimi, süreç tanımlama, organizasyonel eğitim, süreç iyileştirme süreçleri şeklinde organize edilebilir.

- ❖ Süreç Tanımlama Süreçleri: Amacı iş süreçlerinin tutarlı, tekrarlanabilir, performansını destekleyecek süreç kütüphanesini oluşturmaktır. Bu süreç için süreç tanımlama formu, süreç iş çıktıları hazırlama formu gibi şablonlar tanımlanmalıdır.

❖ **Organizasyonel Eğitim:** Amacı; çalışanların kendilerine verilen rolleri verimli ve etkin bir şekilde gerçekleştirebilmeleri için; bilgi beceri düzeylerinin geliştirilmesinin sağlanmasıdır. Bu sürecin sonuçlarının ölçülebilmesi için eğitim kayıt listesi, eğitim sonuçları değerlendirme formu vb. şablonlar tanımlanmalıdır.

**Süreç İyileştirme Süreci:** Amacı; organizasyon süreçleri ve süreç varlıkları ile ilgili kuvvetli ve zayıf alanlar baz alınarak; organizasyon süreç iyileştirme aktivitelerinin planlanması ve uygulanmasıdır.

#### 4.4 Ortak Alt Sistemlerin Tasarımı



Herhangi bir bilgi sistemi tasarlanırken, hemen hemen tüm bilgi sistemlerinde ortak olarak bulunan bazı alt sistemlerin dikkate alınması gerekmektedir. Söz konusu alt sistemler:

### 5.SİSTEM GERÇEKLEŞTİRİMİ

Bu bölümde yani gerçekleştirimde, mantığın bir programlama dili vasıtasıyla eldeki veriler ışığında kodlanması. Tasarım sonucu üretilen süreç ve veri tabanının fiziksel yapısını içeren fiziksel modelin bilgisayar ortamında çalışan yazılım biçimine dönüştürülmesi çalışmasıdır. Her şeyden önce bir yazılım geliştirme ortamı seçilmesidir. (programlama dili, veri tabanı yönetim sistemi, yazılım geliştirme araçları (CASE)).

#### 5.1 Yazılım Geliştirme Ortamları

**Arduino**

#### 5.2 CASE Araç ve Ortamları

Microsoft Project: Gantt Diyagramı çizilmesinde faydalanılmıştır.

Microsoft Word: Dokümantasyonun hazırlanmasında faydalanılmıştır.

ArgoUML: Diyagramların çiziminde faydalanılmıştır.

Microsoft Visio: Kapsam diyagramı ve örgüt yapısı çiziminde yararlanılmıştır.

Smart Draw: Use Case Diyagramları hazırlanırken faydalanılmıştır.

Circuits:Projenin temel tasarım ve testi yapıldı.

### 5.3 Kodlama Stili

5.3.1 Açıklama Satırları Bir yazılım geliştirirken kodların tekrar kullanılabilmesi, başkaları tarafından anlaşılabilmesi için kodlara açıklama satırları koyulur. Bunlar genel olarak Bir paragraf şeklindeyse; /\*Açıklama Açıklama açıklama \*/ Tek bir satır ise; //Açıklama şeklinde ifade edilir.

#### 5.3.1Kod Biçimlemesi

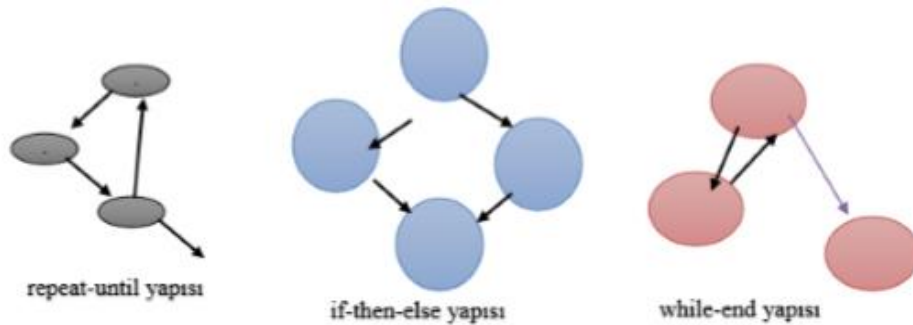
Kodu dosya içinde bulabilirsiniz.

### 5.4.Yapısal Programlama Yapıları

Program kodlarının, okunabilirlik, anlaşılabilirlik, bakım kolaylığı gibi kalite etmenlerinin sağlanması ve program karmaşıklığının azaltılması amacıyla "yapısal programlama yapıları" kullanılarak yazılması önemlidir. Yapısal Programlama Yapıları, temelde, içinde "go to" deyimini bulunmayan, "tek giriş ve tek çıkışlı" öbeklerden oluşan yapılardır. Teorik olarak herhangi bir bilgisayar programının, yalnızca Yapısal Programlama Yapıları kullanılarak yazılabileceği kanıtlanmıştır. Üç temel Yapısal Programlama Yapısı bulunmaktadır:

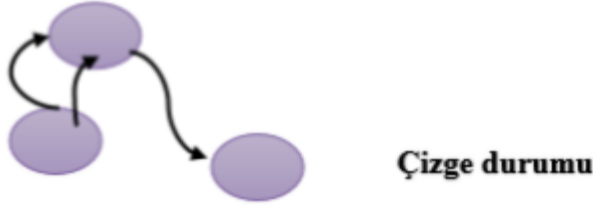
- Ardışıl işlem yapıları
- Koşullu işlem yapıları
- Döngü yapıları

1. Belirli sayıda yinelenen işlemler için kullanılan yapılar (for yapısı)
2. Bir koşula bağlı olarak, sıfır ya da birden çok kez yinelenen işlemler için kullanılan yapılar (while-end yapısı)
3. Bir koşula bağlı olarak, bir ya da daha çok kez yinelenen işlemler için kullanılan yapılar (repeat-until yapısı). Bu yapıların her biri "tek girişli ve tek çıkışlı" yapılardır. Programlar, yalnızca bu yapılar kullanılarak kodlandığında ve uygun açıklama satırları ile desteklendiğinde, program bakımı kolaylaşmakta ve geliştirme hızlanmaktadır.



## 5.5 Program Karmaşıklığı

### 5.5.1 Programın Çizgi Biçimine Dönüştürülmesi



Ölçütteki daireler bir sıradan işlem ya da birden fazla ardışık sıradan işlem durumudur. çizgede bir düğüme dönüştürülür. Ölçütteki oklar ise döngü işlemlerindeki denetimlerin yapıldıkları yerleri belirginleştirmek amacıyla kullanılmıştır

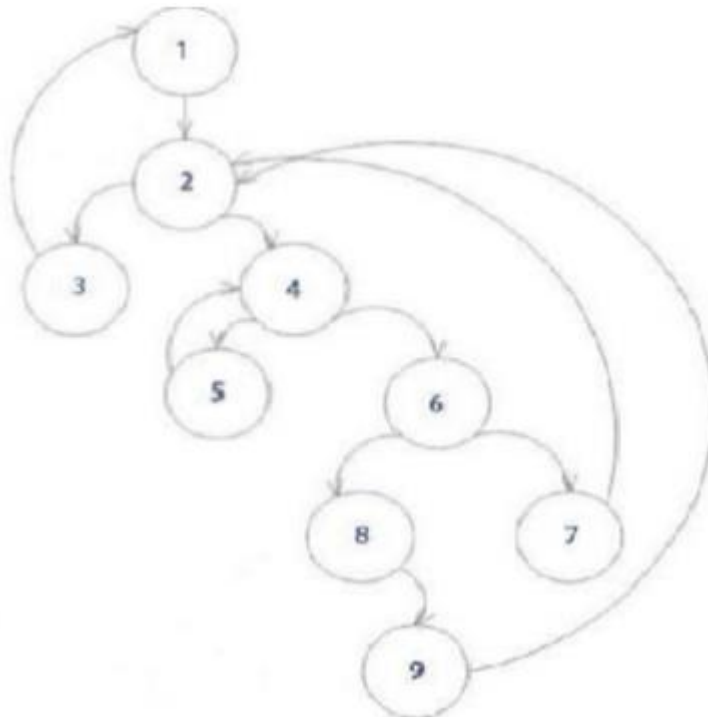
### 5.5.2 McCabe Karmaşıklık Ölçütü Hesaplama

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

**K:** Diyagramdaki kenar çizgi sayısı

**D:** Diyagramdaki düğüm sayısı

**P:** diyagramdaki bileşen sayısı (ana program ve alt program sayısını göstermektedir. Alt program kullanılmadı ise  $p=1.3$  , alt program kullanıldı ise  $p=-1.3$  tür.



**Mccabe Çizgi Biçimi**

## 5.6 Olağan Dışı Durum Çözümleme

Olağan dışı durumlar gerek kod yazım sürecinde gerekse testler sırasında gerçekleşebilir. Projede Helezonik Model kullanıldığından dolayı her aşamada test yapılacağından olağan dışı durum anında çözülebilecektir.

## 5.7 Kod Gözden Geçirme

Bu aşamada gözden geçirme işleminin yapılması diğer bölümlerde yani doğrulama ve geçerleme açısından önemli bir faktördür. Her modül ve alt programlar oluşturulduktan sonra kod gözden geçirilmeleri mutlaka yapılmalıdır. Kod gözden geçirilmelerde programda oluşabilecek hataların %5'inin çözülmesi hedeflenmektedir.

### 5.7.1 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. Burada yanıtı aranan temel soru, programın yazıldığı gibi çalışıp çalışmayacağının belirlenmesidir. Gözden Geçirme çalışmasının olası çıktıları biçiminde özetlenebilir. Burada yanıtı aranan temel soru, programın yazıldığı gibi çalışıp çalışmayacağının belirlenmesidir.

### 5.7.2 Gözden Geçirme Sırasında Kullanılacak Sorular

Yazılım geliştirme ekibinin geliştirdiği kodu gözden geçirmek için bir checklist kullanmak, bu sürecin bir parçasıdır ve uzmanlarca tavsiye edilir. Herhangi bir kod commit edilmeden önce aşağıdaki gibi bir liste ile check edilebilir:

1. Kod doğru bir şekilde build edildi mi? Kaynak kod derlendiğinde hata olmamalı. Kodda yapılan değişikliklerde uyarılar(warning) olmamalı.
2. Kod çalıştırıldığında beklendiği gibi davrandı mı?
3. Kodun sadece çalışırılığına bakılmamalı, kod tasarımı da göz önünde bulundurulmalıdır. Optimizasyon için öneriler takım olarak değerlendirilmelidir.
4. Gözden geçirilen kod anlaşılıyor mu? Gözden geçiricinin kodu anlaması gerekir. Eğer anlaşılmadıysa, gözden geçirme tamamlanmış olmaz veya kod iyi yorumlanabilmiş sayılmaz.
5. Geleneksel kodlama standartlarına uyuldu mu? Değişken isimlendirme, satır başı boşluklar, parantez stilleri vs. takip edilmeli.
6. Telif hakkı bilgisi ve uygun bir başlıkla başlayan kaynak dosya var mı? Her bir kaynak dosyası bu bilgilerle başlamalı, bütün kaynak dosyaları, fonksiyonelliğini anlatan bir dosya içermelidir.
7. Değişken deklarasyonlarına yorum satırları eklenmiş mi? Yorumlar, değişkenlerin görevlerini açıklaması gerekir. Özellikle her bir global değişkenin amacı ve neden global olarak tanımlandığı belirtilmelidir.

8. Sayısal verilerin birimleri açıkça belirtilmiş mi? Sayısal verilerin birimleri yorum satırı olarak belirtilmeli. Örneğin, eğer bir sayı uzunluğu temsil ediyorsa, metre mi feet mi olduğu gösterilmelidir.
9. Bütün fonksiyonlar, metotlar ve classlar dokümante edilmiş mi? Her bir fonksiyon, metot ve class'ın tanımlanmasının üstünde bir iki cümle ile açıklaması yer almalıdır. Amacı vurgulamalı ve tasarım gerekliliklerini işaret etmelidir.
10. Fonksiyonların kullandığı input ve output parametreleri açıkça tanımlandı mı?
11. Karmaşık algoritmalar ve kod optimizasyonları yeterli olacak şekilde açıklanmış mı? Karmaşık alanlar, algoritmalar ve kod optimizasyonları için yeterince yorum satırı eklenmelidir. Öyle ki, diğer geliştiriciler kodu anlayabilmeli ve kalınan yerden devam ettirebilmelidir.
12. Kodun çeşitli yerlerinde yorum satırlarıyla açıklamalar var mı? Kodun çeşitli yerlerinde yorum satırlarıyla açıklamalar olmalı. "Ölü Kod"lar çıkarılmalı. Eğer geçici bir kod bloğu ise neden tanımlandığı belirtilmeli.
13. Koddaki eksik işlevsellikler veya çözülmemiş sorunlar yorum satırlarında ifade edilmiş mi? Bu ifadeler eksikleri ve yapılacakları açıklamalıdır. Sonradan arandığında bulunabilmesi için de ayırıcı bir işaretleyici kullanılmalı, örneğin TODO.
14. Her zaman bir fonksiyonun döndürebileceği hatalar düzün bir şekilde handle edilmeli. Fonksiyonun üreteceği her bir sonuç düşünülmeli, her durum kontrol edilmeli ve kodun kalan kısmının yürütülmesini etkileyen hatalar yakalanmış olmalıdır.
15. Alınan hatalardan sonra tüm kaynaklar ve hafıza temizlenip serbest bırakılıyormu? Bundan emin olunmalı. Bir hata meydana geldiğinde, dosya, soket ve veritabanı bağlantı objeleri gibi tüm objeler dispose edilmeli.
16. Exception'lar uygun bir şekilde yakalanıyor mu? Eğer fırlatılan exception kodun devamında kullanılıyorsa, bu fonksiyon devamında düzgün bir şekilde handle edilmeli veya yakalanmalı.
17. Tüm global değişkenler thread-safe olmalı. Eğer global değişkenlere birden fazla thread ile erişiliyorsa, kodun çalışmasını değiştirebilir veya sekronizasyon mekanizmasını engelleyebilir. Yine benzer bir şekilde olarak bir veya daha fazla thread ile erişilen objeler varsa, üyeler korunmalıdır.
18. Tespit edilen hata kodun başka yerlerini de etkiliyor mu, kontrol edilmeli? Hatanın tüm ekranlarda giderildiğinden emin olunmalı.
19. Kodun değişiklik yapılan yerlerinde, eski halini ve neden yapıldığını mutlaka açıklama olarak eklenmelidir.

20. Kodda yapılmış yorumlar değerlendirilmeli, eğer yorumun uygun/doğru olmadığı düşünülüyorsa geliştirici ile görüşülmeli ve konu tartışıldıktan sonra çözüme kavuşturulmalıdır.

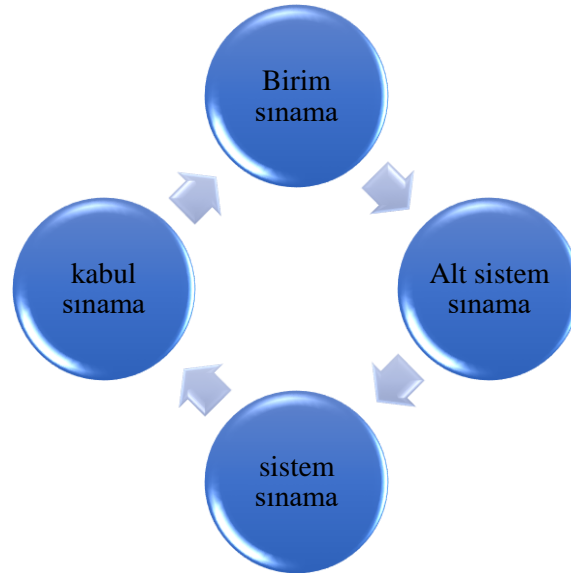
## 6. DOĞRULAMA VE GEÇERLEME

### 6.1 Giriş Doğrulama,

yazılımın yaşam döngüsü boyunca her aşamada bir önceki aşamadaki gereksinimlere uygunluğunu denetleme işlemidir. Geçerleme ise geliştirme işleminin sonunda yazılımın gereksinimlere uygunluğunu, yani kendinden beklenenleri karşılayıp karşılamadığını test etme işlemidir. D&G, yazılımın istenen görevleri doğru şekilde yerine getirip getirmediğini belirlemek, istenmeyen herhangi bir işlem yapmadığından emin olmak ve kalite ve güvenilirliğini ölçmek amacıyla yazılımı kapsamlı bir şekilde test

### 6.2 Sınama Kavramları

Sınama ve Bütünleştirme işlemlerinin bir strateji içinde gerçekleştirilmesi, planlanması ve tekniklerinin seçilmesi gerekmektedir. Sınama işlemleri dört ana sınıfta incelenebilir:



#### 6.2.1 Birim Sınama

Bağılı oldukları diğer sistem unsurlarından tümüyle soyutlanmış olarak birimlerin doğru çalışmalarının belirlenmesi amacıyla yapılır.

### 6.2.2 Alt-Sistem Sınama

Alt-sistemler modüllerin bütünleştirilmeleri ile ortaya çıkarlar. Yine bağımsız olarak sınamaları yapılmalıdır. Bu aşamada en çok hata ara yüzlerde bulunmaktadır. Bu yüzden ara yüz hatalarına doğru yoğunlaşmalıdır.

### 6.2.3 Sistem Sınaması

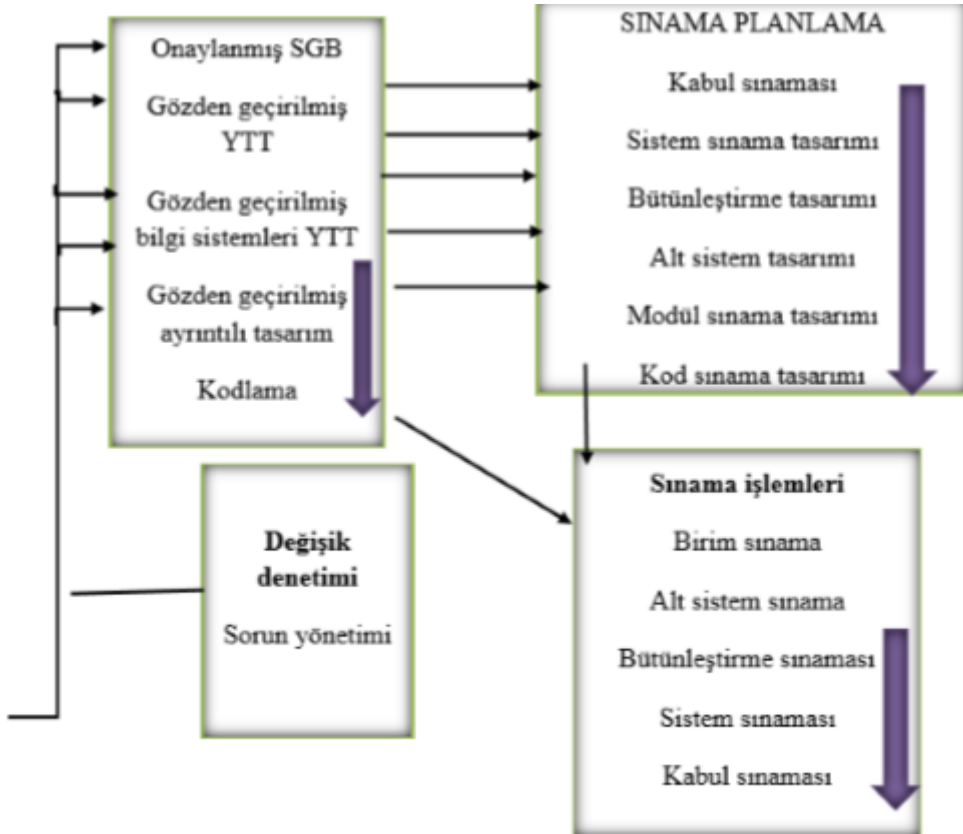
Üst düzeyde, bileşenlerin sistem ile olan etkileşiminde çıkacak hatalar aranmaktadır. Ayrıca, belirtilen ihtiyaçların doğru yorumlandıkları da sınanmalıdır.

### 6.2.4 Kabul Sınaması

Çalıştırılmadan önce sistemin son sınamasıdır. Artık, yapay veriler yerine gerçek veriler kullanılır. Bu sınama türü alfa sınaması veya beta sınaması olarak ta bilinir.

### 6.3 Doğrulama ve Geçerleme Yaşam Döngüsü

Gerçekleştirim aşamasına kadar olan süreçlerde doğrulama ve geçerleme işlemlerinin planlaması yapılır. Planlama genellikle; Alt sistem, Bütünleştirme, Sistem ve Kabul sınamalarının Tasarımlarını içerir. Gerçekleştirim aşamasının sonunda ise söz konusu plan uygulanır.



Doğrulama ve geçerleme işlemleri yazılım üretim yaşam döngüsünün tüm süreçlerinde ve bu süreçlere koşut olarak sürer. Gerçekleştirim aşamasına kadar olan süreçlerde doğrulama ve geçerleme işlemlerinin planlaması yapılır. Planlama, genel olarak, birim, alt sistem, bütünleştirme, sistem ve kabul sınamalarının tasarımlarını içerir. Gerçekleştirim aşamasının sonunda ise söz konusu planlar uygulanır.



## 6.4 Sınama Yöntemleri

Her yazılım Mühendisliği ürünü iki yoldan sınanır:

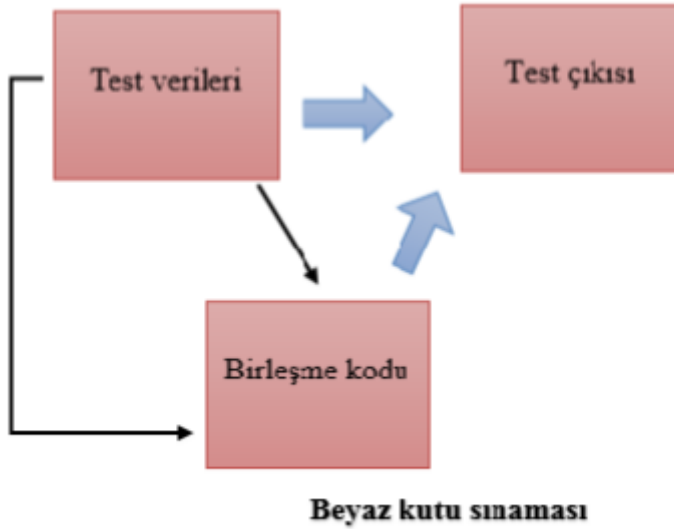
Kara Kutu Sınaması (Black-Box testing ): Sistemin tümüne yönelik işlevlerin doğru yürütüldüğünün testidir. Sistem şartnamesinin gerekleri incelenir.

Temel yollar Sınaması: İç işlemlerin belirtilmelere uygun olarak yürütüldüğünün bileşenler tabanında sınanmasıdır.

### 6.4.1 Beyaz Kutu Sınaması

İç işlemlerin belirtilmelere uygun olarak yürütüldüğünün bileşenler tabanında sınanmasıdır. Beyaz kutu sınanmasında şu noktalara dikkat edilmelidir:

- Bütün bağımsız yolların en az bir kez sınanması gerekir.
- Bütün mantıksal karar noktalarında iki değişik karar için sınamalar yapılır.
- Bütün döngülerin sınır değerlerinde sınanması
- İç veri yapılarının denenmesi



### 6.4.2 Temel Yollar Sınaması

Daha önce çevrimsellik karmaşıklığı konusunda gördüğümüz hesap yöntemi ile bir programdaki bağımsız yollar bulunduğundan sonra, bu kadar sayıda sınamaya yaparak programın her birimini bir şekilde sınamalara dahil etmiş oluruz. Bağımsız yolların saptanması için önce, program çizgisel bir biçime çevrilir. Bunu yapmak için ise, program iş akış şemaları diyagramları iyi bir başlangıç noktasıdır. Bir Program İş Akış şeması

## 6.5 Sınama ve Bütünleştirme Stratejileri

Bu testlerin yapılma amacı sistemde oluşan hataların tespiti ve sistemin dayanıklılığını test edilmesi içindir. Yukarıdan-aşağı ve aşağıdan-yukarıya stratejileri bütünleştirme yöntemine bağlıdır.

### 6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme

Bu işlem şu adımlarda gerçekleşir.

- ✓ En üst düzeylerinin sınanması
- ✓ Aşağıya doğru olan düzeyleri,
- ✓ İlgili modüllerin takılarak sınanmaları söz konusudur.

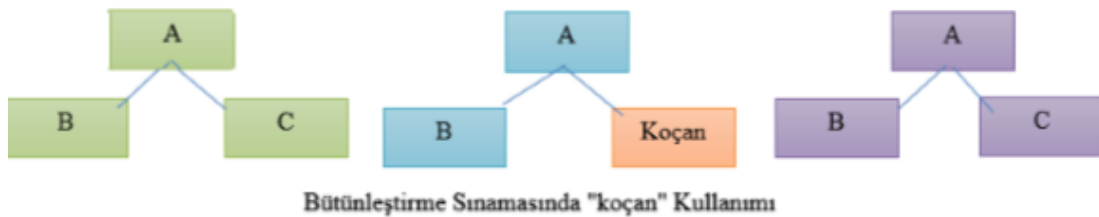
#### Üst bileşen

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.

#### Alt bileşen

Alt bileşenler ise bu stratejiye göre henüz hazırlanmış olamazlar. Bunların yerine üst bileşenin sınaması için kullanılmak üzere 'koçan' programları yazılır. Koçanlar, bir alt bileşenin, üst bileşen ile ara yüzünü temin eden, fakat işlevsel olarak hiç bir şey yapmayan, boş çerçeve programlarıdır. Üst bileşenin sınanması bittikten sonra bu koçanlar, içleri doldurularak kendi kodlama ve birim sınama işlemlerini tamamladıktan sonra üst bileşen ile yeniden sınanırlar Alt bileşen ve üst bileşen arasındaki ilişkiyi koçan sağlar.

**Koçan:** Bir alt bileşenin, üst bileşen ile ara yüzünü temin eden, fakat işlevsel olarak hiçbir şey yapmayan çerçeve programlardır.



### 6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme

Önceki yöntemin tersine uygulama yapılır.

- ✚ Önce en alt düzeydeki işçi birimler sınanır ve bir üst düzey ile sınanması gerektiğinde bu düzey bir sürücü ile temsil edilir.
- ✚ Bu kez kodlama, bütünleştirme ve sınama, aşağı düzeylerden yukarı düzeylere doğru gelişir

## 6.6 Sınama Planlaması

Gantt Grafiği	Görev Adı	Başlangıç	Bitiş	Gantt Chart Timeline																	
				Şub '17	27 Şub '17	13 Mar '17	27 Mar '17	10 Nis '17	24 Nis												
				C	S	C	Ç	P	P	P	C	S	C	Ç	P	P	C	S	C	Ç	
	1	Sınama Planının Hazır Olması	Per 16.03.17	Cum 17.03.17																	
	2	Alt Sistem Sınama Planı	Cmt 18.03.17	Çar 22.03.17																	
	3	Modül Sınama Planı	Cmt 25.03.17	Per 30.03.17																	
	4	Bütünleştirme Sınama	Paz 2.04.17	Cum 7.04.17																	
	5	Kullanıcı Sınama	Cmt 8.04.17	Sal 25.04.17																	

## 6.7 Sınama Belirtileri Öncesi

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.

## SONRASI

Sınama işlemi sonrasında bu belirtilere,

- ✚ Sınamayı yapan,
- ✚ Sınama tarihi,
- ✚ Bulunan hatalar ve açıklamaları türündeki bilgiler eklenerek sınama raporları oluşturulur.

### 6.7.1Yaşam Döngüsü Boyunca Sınama Etkinlikleri



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.
- Şekilsel Hatalar: Heceleme hatası gibi önemsiz hatalardır.

## 7.BAKIM

### 7.1 Giriş:

Bilgisayar tabanlı sistemlerin tasarlanıp geliştirilmesinde ve kullanıcıya teslim edilmesinden sonra bakım (maintenance) aşaması başlar. Sistemin donanım öğelerinin bakım tutumu, temizleme, eskime ya da arızalanmış parçaların değiştirilmesi şeklinde yapılır veya tüm donanımın ögesi yenisi ile değiştirilir. Yazılım öğelerinde bu eskime yada arızalanma durumu yoktur. Zaman içinde ortaya çıkan yeni isteklerin karşılanması veya bulunan hataların giderilmesi çalışmaları yazılım bakımının temelini oluşturur. Yazılım bakımı yazılım geliştirilmesinden daha fazla zaman ve iş gücü gerektirebilir.

Yazılım bakımı bir aracın bakımı gibi periyodik olmayıp gelişen koşullara göre ortaya çıkar. Örneğin daha önce hazırlanmış bir yazılım değişen teknoloji nedeniyle yeni bilgisayar mimarileri ve işletim sistemlerine uyarlanması önemli bir bakım işidir.

Değişiklik; bilgisayar tabanlı sistemler için kaçınılmaz bir gerçektir. O halde değişiklikleri değerlendirmek, denetim altında tutmak ve uygulamak için bir düzen geliştirmeli ve ona uyulmalıdır. IEEE 1219 Standart For Software Maintenance, bu konuda uygulanabilecek yöntemleri, araç ve teknikleri anlatmaktadır. Yazılım bakımıyla ilgili planlar ve içerikleri de IEEE/EIA 12207 standardı içinde yer almaktadır. Bu bölümde yazılım bakımının nasıl yapılacağı hakkında temel bilgiler verilecektir.

### 7.2 Yazılım Bakımı

Yazılım sisteminin geliştirilmesi, yazılım ürününün müşteriye teslimi ve kullanılmaya başlanması ile tamamlanmış olmaktadır. Kullanım süresinde de bakımı ve korunması, onarılması ve geliştirilmesi gerekmektedir. Yazılım bakımı ve onarımı (software maintenance) deyimini yerleşmiş olmasına karşın, yazılım göz ile görülmediği ve donanım gibi aşınmadığı için, uygun bir terim değildir. Burada söz konusu işlemler; sonradan görülen hataların düzeltilmesi, yazılımın iyileştirilmesi (uyarlanması) ve geliştirilmesi şeklindedir. Geniş bir programın kullanımı sırasında, sınama aşamasında bulunmamış olan çeşitli işlem, yetenek ve tasarım hataları ortaya çıkabilmektedir. Bu hatalar, işlem sırasında kilitlenme, kesilme, yavaş çalışma, anormal işlem vb. olaylarla kendini göstermektedir. Ayrıca, yazılımın yeteneğini iyileştirmek ve kullanımını kolaylaştırmak üzere programa bazı eklemeler gerekebilmektedir. Donanımın, işletim sisteminin ya da verinin değiştirilmesi halinde de yazılımın yeni ortama uydurulması yoluna gidilmektedir. Bu durumda, birkaç modül üzerinde değişiklik yapmak gerekmektedir. Ayrıca, günün koşullarına göre yeterli bulunmayan bir yazılımın; gereksinim analizi, tasarım; tamamlama ve sınama basamakları halinde yenidengeliştirilmesi ve böylece onarılması da söz konusudur. Yazılım bakımı konusundaki işlerin %21'inin hata düzeltme, %25'inin iyileştirme, %50'sinin uyarlama ve %4'ünün diğer durumlarda olduğu bildirilmektedir.

Yazılım bakımı ve korunması işlemi, kullanışa hazır oluşundan itibaren başlatılmakta ve kullanımdan kaldırılıncaya kadar (10-15 yıl) sürmektedir. Önce, yazılım konfigürasyonu (software configuration)) oluşturularak, gerekli düzeltme ya da değişiklikler yerine getirilmektedir.

## **Bakım Maliyetlerinin Azaltılması**

Bakım ve onarım giderini ve en aza indirmek için, yazılım ürününün “bakım ve onarıma elverişli” nitelikte oluşturulması gerekmektedir (maintainability). Bu nitelik; yazılım kolay anlaşılabilir, düzeltilebilir, uyarlanabilir ve geliştirilebilir özelliklerde tasarlanmış olmasıyla sağlanabilmektedir.

Bunu için de ;

- Yetenekli ve deneyimli yazılım mühendisleri görevlendirmek
- Anlaşılabilir bir sistem yapısı ve kolay işletilebilir bir sistem tasarlamak
- Standart programlama dilleri, işletim sistemleri kullanmak ve belgeleri standart biçimde düzenlemek
- Test programlarından yararlanmak
- Tasarım aşamasında, hata bulma ve düzeltme kolaylıkları sağlamak gerekmektedir.

## **Bakım ve Onarıma Elverişlilik**

Yazılımın bakım ve onarıma elverişliliğini ölçmek, diğer kalite faktörlerinde olduğu gibi, ancak dolaylı olarak ve çok sayıda başka özelliklere dayanarak gerçekleştirilmektedir. Bu özellikler; sorunun tanımı, yönetiminde gecikme, gerekli araçların derlenmesi, sorunun analizi, spesifikasyonlarının değiştirilmesi, düzeltme, sına ve gözden geirme süreleri olarak alınabilir. Bu bilgiler, yöneticiye de yeni teknikler ve araçlar kullanma ihtiyacını göstermektedir.

Yazılım bakım ve onarma elverişliliği; yazılımın diğer kalite faktörlerinden olan,

- Sına kolaylığı
- Basitlik
- Değiştirilebilirlik
- Taşınabilirlik
- Güvenirlik
- Esneklik

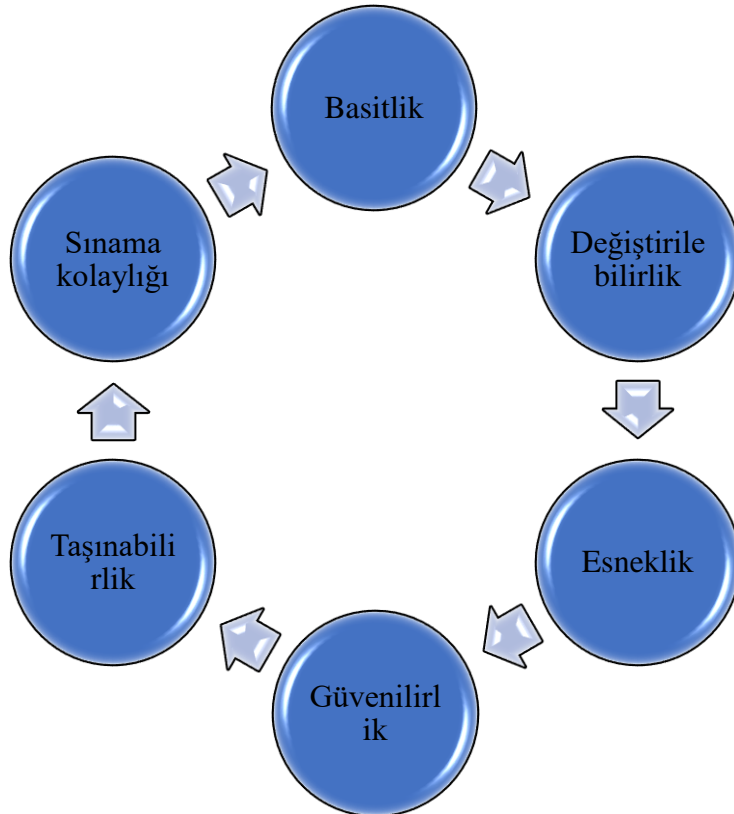
Özelliklerinin bir bileşkesi olarak ortaya çıkmaktadır. Bu nedenle ölçümünde bu faktörlerin varlığını denetlemek yolunda kullanılmaktadır.

## **Yazılım Bakım ve Onarımı Elverişliliğinin Önemi**

Yazılımın bakımı onarıma elverişliliği, he geliştirme basamağının gözden geçirilmesi sırasında önemle dikkate alınmalıdır. Gereksinim analizi basamağında; yazılımın gelecekte

büyütülmesi olasılığı, taşınabilirliği, sistem arabirimleri üzerinde durulmalıdır. Tasarım basamağında veri tasarımı- mimari tasarım ve işlemsel tasarım; değiştirme kolaylığı, modülarite ve işlevsek bağımsızlık bakımlarından gözden geçirilmelidir. Kaynak programın dayanıklılığı ve program giriş belhgelere incelenmelidir. Sınama sırasında alt programların bakım ve onarıma elverişliliği de aranmalıdır. En son olarak da yazılım konfigürasyonu gözden geçirilmelidir (configuration review). Burada; yazılım düzeninin bütün öğelerinin tamam; anlaşılabilir ve bakım- onarım denetiiminde uygun biçimde kütüklere ayrılmış bulunduğu saptanmaktadır.

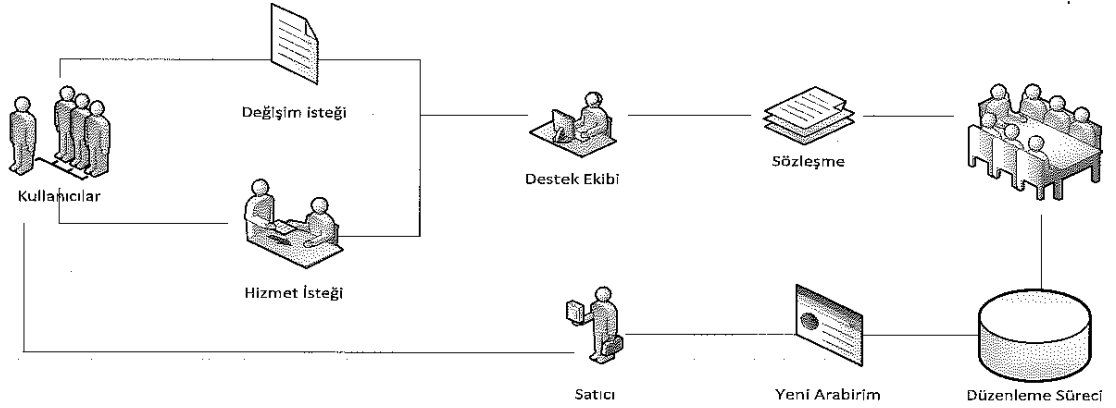
Bakım ve onarım (koruma) işlemlerine başlamadn önce; bu amaçla bir örgüt kurulması, rapor düzenleme ve değerlendirme yöntemlerinin belirlenmesi, her görev için uygulanacak işelrin sıralı olarak tanımlanması, bir kayıt sistemi düzenlenmesi ve değerlendirme kriterlerini konulması gerekmektedir. Bu esaslara göre yazılım bakıma alınmakta ve kullanıcı ile de bağlantı kurularak denetlenmekte, düzeltilmekte ve tekrar gözden geçirilmektedir.



#### Yazılım Bakım:

- Tanım
- Bakım Süreç Modeli

- Sorun tanımlama süreci
- Çözümleme süreci
- Tasarım süreci
- Gerçekleştirim Süreci
- Sistem sınaama Süreci
- Kabul sınaama süreci
- Kurulum süreci



**Şekil-7.4 Yazılım bakım işlemi**

### 7.3 Tanım

Yazılım bakımı, müşteri ile geliştirici arasında yapılan sözleşmeye göre bazen yazılım geliştirme sürecinde yer almamaktadır. Oysa günümüzdeki pek çok uygulama alanı değişen istelere sahiptir. Bu nedenle yazılım geliştirip kullanıma sunduktan sonra dahi büyük miktarda değişiklik ve hata düzeltme istekleri oluşabilmektedir. Bakım, işleme alınan yazılımın sağlıklı olarak çalışması ve ayakta tutulabilmesi için yapılması gereken çalışmalar bütünü olarak tanımlanır. Uygulamada çalışan bir yazılımın üç tür bakım gereksinimi bulunmaktadır.

Bakım yazılım yaşam döngüsünün en önemli ve en maliyetli aşamalarından biridir. Bakım süreci, yazılım yaşam döngüsünde “buzdağının görünmeyen kısmı” olarak adlandırılır. Bakım maliyetleri, zaman zaman üretim maliyetlerinin %60’ını geçer.

Bu bölümde, yazılım bakımı ile ilgili kavramlar açıklanmaktadır. Bu kavramların açıklanmasında IEEE 1219 standardı baz alınmıştır.

**Düzeltilici Bakım:** Yazılım testi her zaman tüm kusurların bulunmasını ve giderilmesini sağlamaz. Çalışan bir yazılımda her an hata bulunma olasılığı vardır. Bir kısım yazılım kusuru ancak kullanım sırasında ortaya çıkar. Bu kusurlar giderilmesi için geliştiriciye bildirilir. Kusurun kaynaklanma sebebini araştırmaya ve gidermeye yönelik işlere düzeltici (corrective) bakım adı verilir. Kullanım sırasında bulunan ve raporlanan yazılım kusurlarının giderilmesi, hatanın önemine bağlı olarak, ya hemen yapılır ya da birkaç



tanesi ile beraberce ele alınıp düzeltmeler uygulanır ve yeni bir sürüm çıkarılır. Bu bakım geliştiricinin bakım aşamasında mutlaka yürütmesi gereken bir etkinliktir.

**Uyarlayıcı Bakım:** Bilgi işleme dünyasındaki hızlı gelişmeler ve teknolojik gelişmeler nedeniyle yazılımın yeni donanıma, işletim sistemlerine, bunların yeni sürümlerine, yeni uçbirimlere göre uyarlanması, sürümün yükseltilmesi ve güncelleştirilmesi işlerine uyarlayıcı (adaptive) bakım denmektedir.

**İyileştirici Bakım:** Yazılım geliştirilip test edilere başarılı bir şekilde kullanıcıya sunulduktan sonra, yeni işlevler eklenerek var olanlara başarıyı ve verimi artırıcı düzeltmeler yapmak iyileştirici (perfective) bakım işleri içinde yer alır. Bu şekilde yazılımın yeni sürümler, ortaya çıkar ve kullanıcının hizmetine sunulur.

**Önleyici Bakım:** Yazılımın gelecekte uygulanabilecek değişikliklere daha iyi bir temel oluşturması, bakılabilirliğinin ve güvenilirliğinin artırılması için ön tedbir niteliğindeki işlemler önleyici(preventive) bakım kapsamına girmektedir.

#### **7.4. Bakım süreç modeli**

IEEE 1219 standardı tarafından önerilen bakım süreç modeli şekli belirtilen şablonu kullanarak bakım süreçlerini tanımlamaktadır.

Bakım süreç modelinin süreçleri:

- Sorun tanımlama/sınıflandırma,
- Çözümleme,
- Tasarım,
- Gerçekleştirim,
- Sistem sınama,
- Kabul sınama,
- Kurulum.

biçimindedir. Görüldüğü gibi bakım süreci, yazılım yaşam döngüsü çekirdek adımlarının bir anlamda yinelenmesinden oluşmaktadır. Bu yinleme yalnızca değişiklik isteklerinin var olan koda aktarılması amacıyla yapılmaktadır. Bakım süreçleri aşağıda açıklanmaktadır.

## 8.SONUÇ

Sonuç olarak emlakçının satıcı ve alıcılarla alış verişini en rahat biçimde yapabilmesi için geliştirilmiş bir simülasyondur.

## 9.KAYNAKLAR

<http://usluer.org/c-sharp-emlak-programi/>

[https://www.youtube.com/watch?v=J\\_aWhaJ1wto](https://www.youtube.com/watch?v=J_aWhaJ1wto)

<https://social.msdn.microsoft.com/Forums/tr-TR/8133e264-6beb-477f-879e-ba3d44fc6218/sql-sorgusu-parametre-kullannca-almyor?forum=csharptr>

<https://www.youtube.com/watch?v=5T6XQarpXgQ&list=PLh9ECzBB8tJNnowfMHINA00u8cJboZzNt>

