



FIRAT ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
Yazılım Mühendisliği Bölümü

YMH217 – Nesne Tabanlı Programlama Dersi
Proje Uygulaması ve Dokümantasyonu

Sinema Bileti Satış Otomasyonu

Geliştiren

Muhammed Can GELERİN

No:16541518

Şube:B

İçindekiler

1. TANITIM.....	4
1.1 Projenin Amacı	4
1.2 Projenin Kapsamı	4
1.3 Tanımlamalar ve Kısaltmalar	4
2. PLANLAMA.....	5
2.1 Giriş	5
2.2 Projenin Plan Kapsamı	6
2.2.1 Proje Kaynaklar	7
2.2.2 İnsan Kaynakları	7
2.2.3 Donanım Kaynakları	7
2.2.4 Yazılım Kaynakları	8
2.3 Proje Zaman ve İş Planı.....	9
2.4 Proje Ekip Yapısı.....	10
2.5 Önerilen Sistemin Teknik Tasarımı.....	10
2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları	11
2.7 Proje Standartları, Yöntem ve Metodolojiler.....	11
2.7.1 Helezonik Modelin Üstün Yönleri	12
2.8 Kalite Sağlama Planı	12
2.9 Konfigürasyon Yönetim Planı	13
2.10 Kaynak Yönetim Planı.....	13
2.11 Eğitim Planı	13
2.12 Test Planı	14
2.13 Bakım Planı	14
3. SİSTEM ÇÖZÜMLEME	14
3.1 Mevcut Sistem İncelemesi.....	14
3.2 Var olan ve Gereksenen Sistem.....	15
3.2.1 Örgüt Yapısı	15
3.2.2 İşlevsel Model	16
3.2.3 Bilgi Sistemleri/Nesneler	20
3.2.4 Veri Modeli	20

3.3	Arayüz (Modül) Gerekleri	21
3.3.1	Yazılım Arayüzü	21
3.3.2	Kullanıcı Arayüzü	21
3.3.3	Personel Paneli	22
3.3.4	Yönetici Arayüzü	24
3.4	Belgeleme Gerekleri	25
3.4.1	Geliştirme Sürecinin Belgelendirmesi	25
3.4.2	Eğitim Belgeleri	25
3.4.3	Kullanıcı El Kitapları	25
4.	SİSTEM TASARIMI	26
4.1	Genel Tasarım Bilgileri	26
4.1.1	Genel Sistem Tasarımı	26
4.1.2	Varsayımlar ve Kısaltmalar.....	27
4.1.3	Sistem Mimarisi	28
4.1.4	Dış Arabirimler	31
4.1.5	Veri Modeli	33
4.1.6	Testler.....	33
4.1.7	Performans	33
4.2	Veri Tasarımı.....	33
4.2.1	Tablo Tanımları.....	34
4.2.2	Veri Tanımları	34
4.3	Süreç Tasarımı.....	34
4.3.1	Genel Tasarım	34
4.3.2	Modüller.....	35
5.	SİSTEM GERÇEKLEŞTİRİMİ	37
5.1	Giriş	37
5.2	Yazılım Geliştirme Ortamları	38
5.2.1	Veri Tabanı Yönetim Sistemleri	38
5.3	Kodlama Stili.....	45
5.3.1	Açıklama Satırları	46
5.3.2	Kod Biçimlemesi.....	47
5.3.3	Anlamlı İsimlendirme	47
5.3.4	Yapısal Programlama Yapıları (YPY)	48

5.4	Program Karmaşıklığı.....	49
5.4.1	Programın Çizge Biçimine Dönüştürülmesi	50
5.4.2	McCabe Karmaşıklık Ölçütü Hesaplama	50
5.5	Olağan Dışı Durum Çözümleme	52
5.5.1	Olağandışı Durum Tanımları	52
5.5.2	Farklı Olağan Dışı Durum Çözümleme Yaklaşımları.....	52
5.6	Kod Gözden Geçirme	53
5.6.1	Gözden Geçirme Sürecinin Düzenlemesi	53
5.6.2	Gözden Geçirme Sırasında Kullanılacak Sorular.....	54
6.	DOĞRULAMA VE GEÇERLEME.....	56
6.1	Giriş	56
6.2	Sınama Kavramları	57
6.3	Doğrulama Ve Geçerleme Yaşam Döngüsü.....	57
6.4	Sınama Yöntemleri	58
6.4.1	Beyaz Kutu Sınaması	59
6.4.2	Temel Yollar Sınaması.....	59
6.5	Sınama Ve Bütünleştirme Stratejileri	62
6.5.1	Yukarıdan Aşağı Sınama Ve Bütünleştirme	62
6.5.2	Aşağıdan Yukarıya Sınama Ve Bütünleştirme.....	63
6.6	Sınama Planlaması.....	64
6.7	Sınama Belirtileri.....	65
6.8	Yaşam Döngüsü Boyunca Sınama Etkinlikleri	66
7.	BAKIM.....	67
7.1	Giriş	67
7.2	Kurulum.....	68
7.3	Yerinde Destek Organizasyonu	69
7.4	Yazılım Bakımı.....	69
7.4.1	Tanım	72
7.4.2	Bakım Süreç Modeli.....	74
8.	KAYNAKÇA	88

1. Tanıtım

Projenin Adı: Sinema Bileti Satış Otomasyonu

1.1 Projenin Amacı

Projenin en büyük amacı sinema gişelerinde işlemlerin daha kısa sürmesini sağlamaktır. Ayrıca sinema biletinin isim, tarih, koltuk numarasına göre kesilerek oluşabilecek sorunların önüne geçilmesi sağlanır. Proje kapsamında hem personellerin işi kolaylaşır hem de müşterinin memnuniyeti sağlanmış olur. Ayrıca müşteri perdeye göre koltuğunu kendi seçebilmesine olanak sağlanır.

1.2 Projenin Kapsamı

Projem müşteri, personel ve yetkili personel üzerine kurulmuştur. Müşteri için zaman kaybını azaltmak, perdeye göre istediği yerden koltuk alabilme ve oluşabilecek sorunların en aza indirmek müşteri memnuniyetini sağlayacak; personel için işlerin kolaylaşması ve daha rahat bir ortamın oluşması da personelin işlerini kolaylaştıracaktır. Ayrıca film ekleyip kaldırma, salon numarasına göre atama yapma gibi özelliklerle de programın devamlılığı sağlanacaktır. Bilet bu özelliklere göre kesildiği için müşterinin hangi salonda film izlemesi gerektiğini kolayca bulabilecektir. Ayrıca ücretlendirme ekleyip kaldırabilme ile öğrenci ve tam biletlerin kesim ücretleri farklılık gösterir.

1.3 Tanımlamalar ve Kısaltmalar

Salon No: Gösterime giren filmin kaçınıcı salonda yayınlandığını gösteren numara

Filmin Adı: Gösterime giren filmin adı

Film Ekle/Kaldır: Gösterime girecek veya gösterimi biten filmin eklenmesi veya kaldırılması

Salon Ekle/Kaldır: Gösterime girecek veya gösterimi biten filmin salon numaralarına göre atanması veya kaldırılması

Koltuk No: Salon içindeki koltukların numarası

Boş/Dolu Koltuklar: Müşterinin koltuk seçimi yaparken boş veya dolu koltukları görerek seçimini kolaylaştırmasını sağlayan ikonlar

Bilet İptal: Kullanıcının bileti iptal etmek istemesi ile sistemden biletin alınmamış gibi olması sağlanır

Perde: Sinema salonunda filmin yansıtıldığı alan

Ücret: Sinemadaki filmin ücreti

Gösterim Tarih/Saat(Seans): Filmin gösterim tarih ve saati

Bilet Kesim Tarih/Saat: Biletin kesim tarih ve zamanı

Ücretlendirme Ekle/Kaldır: Ücretlendirmeyi sınıflara göre ayırıp farklı ücretlendirmeler yapılmasını sağlar

2. Planlama

2.1 Giriş

Sinema otomasyonu yapmaya başlanmadan önce planlama yapılması şarttır. Yazılım gelişim sürecinin ilk adımı planlamadır. Bu bölümde yazılımın tüm gereksinimleri planlanır. Proje planlama aşamasında yapılan işlemler;

Proje kaynaklarının belirlenmesi,

Proje maliyetinin kesinleştirilmesi,

Proje ekip yapısının oluşturulması,

Ayrıntılı proje planının yapılması,

Projenin izlenmesidir.

Proje planı tüm proje süresince kullanılıp, müşterinin isteği ve projenin gelişimi göz önünde bulundurularak güncellenecek bir belgedir. Bu belge projenin temel taşlarını oluşturur. İyi bir planlama projenin en az hata ile başarılı sonuçların elde edilmesi için en gerekli bölümdür. Projede hataların az olması maliyeti azaltır ve aynı zamanda maksimum kazanç sağlanır. Planlama sürecinde müşterinin istekleri de oldukça önemlidir.

2.2 Projenin Plan Kapsamı

Maliyet Kestirim Dokümanı

Adı: Sinema Bilet Satış Uygulaması

Ölçüm Parametresi	Sayı	Ağırlık	Toplam
Kullanıcı Girdi Sayısı	8	5	40
Kullanıcı Çıktı Sayısı	4	4	16
Kullanıcı Sorgu Sayısı	4	6	18
Kütük	3	3	9
Ana İşlev Nokta Sayısı(AİN)			83

Teknik Karmaşıklık Faktörü (TKF)	Puan
1. Sistem, güvenilir yedekleme ve kurtarma gerektiriyor mu?	2
2. Veri iletişimi gerektiriyor mu?	1
3. Performans kritik mi?	3
4. Sistem, mevcut ve ağır yükü olan bir işletim ortamında mı çalışacak?	1
5. Sistem, çevirim içi veri girişi gerektiriyor mu?	1
6. Çevirim içi veri girişi, bir ara işlem için birden çok ekran gerektiriyor mu?	0
7. Girdiler, çıktılar ve sorgular karmaşık mı?	0
8. Sistem içi işlemler karmaşık mı?	2
9. Tasarlanacak kod yeniden kullanılabilir mi?	3
10. Dönüştürme ve kurulum, tasarımda dikkate alınacak mı?	5
11. Sistem birden çok yerde yerleşik farklı kurumlar için mi gerçekleştiriliyor?	2
12. Tasarlanan yazılım (projesi dahil) geliştirilecek mi?	5
Teknik Karmaşıklık Faktörü (TFK) Toplam	25

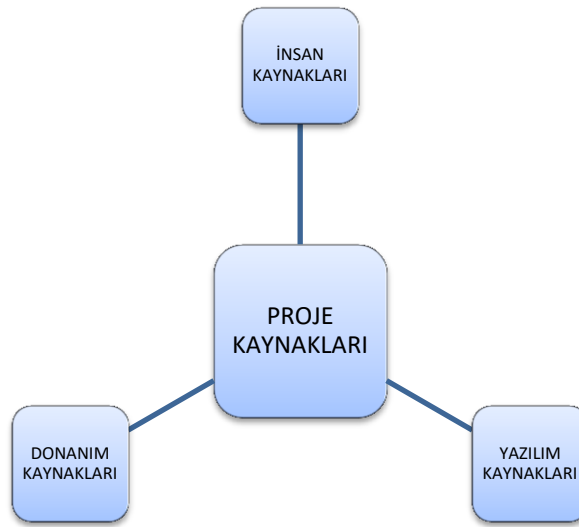
Kullanıcı Girdileri: Modüllerin ve Yetkili Yöneticinin Yaptığı işlemleri ifade etmektedir.

Kullanıcı Çıktısı: Modüller ve Yetkili Yöneticinin yaptığı işlemlerin sonuçlarını ifade etmektedir.

Kullanıcı Sorguları: Çevrimiçi olarak bilgisayara verilen Bir girdi sonucu, yine çevrimiçi olarak bir kullanıcı çıktısı alınması biçiminde tanımlanmaktadır.

Kütükler: Sistem tarafından kullanılan veri tabanı sayısını belirtmektedir

2.2.1 Proje Kaynaklar



2.2.2 İnsan Kaynakları

*Proje Yöneticisi

*Programcı

*Eğitmen (Programın nasıl kullanacağını müşteriye seminer veren kişi)

*Test Ekibi

2.2.3 Donanım Kaynakları

*Kullanıcı bilgisayarı(Pc)

*Masaüstü bilgisayar

*Monitör

*Yazıcı

2.2.4 Yazılım Kaynakları

- * Microsoft Windows İşletim Sistemi
- * Windows 8, Windows 7, Windows XP vb.
- * Adobe Photoshop
- * Python ya da C# (Nesne tabanlı programlama dilleri)

Hesaplamalar:

Ana İşlev Nokta Sayısı(AİN) = 83

Teknik Karmaşıklık Faktörü(TFK) = 25

$$\dot{IN} = A\dot{IN} * (0.65 * 0.01 * TKF) \Rightarrow \dot{IN} = 83 * (0.65 * 0.01 * 25)$$

$$\dot{IN} = 13,4875$$

$$\text{Satır Sayısı}(S) = \dot{IN} * 30 \Rightarrow \text{Satır Sayısı}(S) = 13,4875 * 30$$

$$\text{Satır Sayısı}(S) = 404,625$$

İş Gücü(K) = a * S^b (a, b: Her bir model için farklı katsayılar alır; S bin türünden satır sayısı.)

$$\text{Yarı Gömülü Proje İçin: İş Gücü}(K) = 3,0 * S^{1,12} \Rightarrow 3,0 * 0.404625^{1,12}$$

$$\text{İş Gücü}(K) = 1.08 \Rightarrow 1 \text{ Kişi Bu Projeyi Yapabilir.}$$

Zaman(T) = c * K^d (c, d: Her bir model için farklı katsayılar alır.)

$$\text{Yarı Gömülü Proje İçin: Zaman}(T) = 2.5 * K^{0.38} \Rightarrow \text{Zaman}(T) = 2,5 * 1,08^{0,38}$$

$$\text{Zaman}(T) = 2,574... \Rightarrow \text{Bu proje 2,5 ay sürecektir.}$$

$$\text{Üretkenlik} = \text{İN} / \text{Kişi} - \text{Ay}$$

$$\text{Kalite} = \text{Hatalar} / \text{İN}$$

$$\text{Üretkenlik} = 13,4875 / 2,5 - 1$$

$$\text{Kalite} = 1 / 13,4875$$

$$\text{Üretkenlik} = 8,99...$$

$$\text{Kalite} = 0,0741427147451..$$

$$\text{Maliyet} = S / \text{İN}$$

$$\text{Satır Sayısı} = \text{İN} * \text{Dil Kodu}$$

$$\text{Maliyet} = 0,404625 / 13,4875$$

$$\text{Satır Sayısı} = 13,4875 * 30$$

$$\text{Maliyet} = 0,03$$

$$\text{Satır Sayısı} = 404,625$$

2.3 Proje Zaman ve İş Planı

Projeye başlamadan önce tüm kesimle (Sinema salonu işletmeleri, sinema personelleri vb.) iletişime geçerek sinema salonu işletmesinin ortak sorunlarının listesini hazırlamak ilk iş olacaktır. Belirtilen istekler doğrultusunda sistemin olurluluğu araştırılacaktır. Bu aşamada tüm isterlerin istekleri iyi anlaşılabilir olarak gerçekleştirilmesi gerekmektedir.

Analiz süreci keşfetme ve geliştirme sürecidir. Gereksinim analizi yapılmalı ve kullanıcının istekleri göz önüne alınmalıdır. Analiz süresi çok uzun tutulmamalıdır. İyi sonuçlar elde etmek için analiz sürecinde dikkatli olunmalıdır.

Mimari tasarım aşaması diğer aşamalara göre daha az zaman alabilir. Çünkü görsel tasarım fazla karmaşık değildir.

Yazılımın kodlanması aşaması ise geliştirici için kritiktir. Çünkü yazılımın kodlanmasında okunula bilirlik ve yazıla bilirlik çok önemlidir. Kodlama sırasında sistemin iyi bilinmesi gerekir. Oluşabilecek hatalara karşı çözüm üretmek amacı için sistemi iyi tanıyan kişiler görev almalıdır.

Kodlama aşaması sonrasında test değerlendirme aşaması gerçekleşmektedir. Oluşan sistemin belirli bir kullanıcıya sunularak test yapılır ve olumsuzluklar üzerinde değerlendirilmede bulunulur. Eksiklik varsa giderilmeye çalışılır. Test aşamasına ayrılan süre de çok olmamalıdır.

Belgelendirme aşamasında projenin detayları raporlanarak yazılım geliştirme süreci tamamlanır.

Son aşama bakım aşamasıdır ve bu aşamanın süresi kesin olarak bilinmez. Sistemde zaman içinde oluşabilecek sorunlara çözümler üretilir.

#	Aşamasını Adı	Haftalar									
		1	2	3	4	5	6	7	8	9	10
1	Giriş										
2	Plan										
3	Sistem Çözümleme										
4	Tasarım										
5	Risk Analizi										
6	Test										
7	Bakım										

2.4 Proje Ekip Yapısı

Bu proje fazla kapsamlı olmadığından ve projenin sadece bir kişi tarafından yapılabileceğinden dolayı projeyi yapan kişi projenin her alanında yer almaktadır. Projeyi planlayacak, tasarlayacak, kodlayacak, test edecek kişi projeyi yapan kişi olacaktır. Eğer yapılamayan nokta olur da projeye yardım eden bir kişi daha gelirse projenin zamanın azalacaktır.

2.5 Önerilen Sistemin Teknik Tasarımı

Sistemim offline olduğu için herhangi bir internet bağlantısına gerek duyulmaz. Windows işletim sistemine sahip; laptop ve masaüstü bilgisayardan programa erişim sağlanabilir.

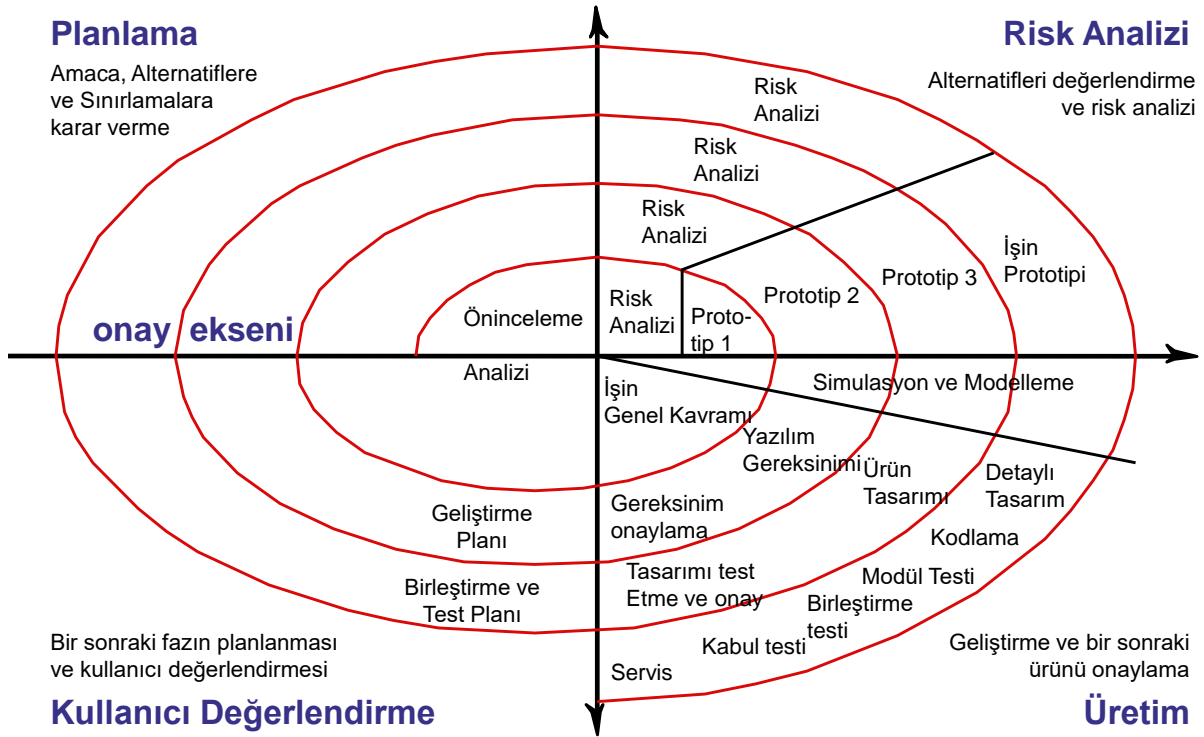
Program nesne tabanlı bir programlama dilinde gerçekleştirilecek ve kullanıcıya kolay kullanım sağlanacaktır.

2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları

- C# kullanıldığı için Visual Studio 2012 ve üst versiyonları seçilebilir.
- Görsel tasarım için Adobe Photoshop kullanılacaktır
- Proje süresince Helezonik (Spiral) Model kullanılacaktır.

2.7 Proje Standartları, Yöntem ve Metodolojiler

Projede Helozonik (Spiral) Model tercih edilmektedir. Bunu tercih etmemin nedeni projeyi müşterinin değerlendirme yaparak daha olumlu sonuçların elde edilmesidir. Böylelikle hem müşteri memnuniyeti sağlanır hem de program içi hata varsa daha kolay fark edinilebilir.



Helezonik model genel olarak art arda tekrarlanan dört aşamadan meydana gelir. Bunlar;

- Amaçların belirlendiği, olası seçeneklerin ve kısıtlamaların değerlendirildiği planlama aşaması
- Diğer yöntemlerde bulunmayan, risklerin tanımlandığı ve olası çözüm yöntemlerinin irdelendiği risk çözümleme aşaması
- Ürünün geliştirildiği mühendislik aşaması
- Geliştirilen ürünün müşteriyle beraber incelendiği değerlendirme aşaması

2.7.1 Helezonik Modelin Üstün Yönleri

Yazılım geliştirme süreç modellerinden; Gelişi Güzel Model, Barok Modeli, Çağlayan Modeli, V Modeli, Evrimsel Model ve Araştırma Tabanlı Model seçenekleri arasından Helezonik Modelin ayrıcalıklarını aşağıdaki gibi açıklayabiliriz.

- **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.
- **Yönetici Bakışı:** Gerek proje sahibi, gerekse yüklenici tarafındaki yöneticiler, çalışan yazılımlarla proje boyunca karşılaştıkları için daha kolay izleme ve hak ediş planlaması yapılır.
- **Yazılım Geliştirici (Mühendis) Bakışı:** Yazılımın kodlanması ve sınanması daha erken başlar.

2.8 Kalite Sağlama Planı

Yazılımın kalite sağlama planında, SPICE (ISO 15504) Modeli referans alınacaktır. Bu modelin kapsamında;

- Yazılımın satın alınması,
- Yazılımın geliştirilmesi,
- Yazılımın sınanması ve iyileştirilmesi,
- Yazılımın işletimi,
- Bakım ve destek süreçleri

Model kapsamı içindedir.

2.9 Konfigürasyon Yönetim Planı

Konfigürasyon yönetimi tasarım ve üretim süreçlerindeki işlemlerin kontrol altında bulundurulması, müşterinin talep ettiği bilgilere anında ve kolay ulaşılması, hataların azaltılması ve verimliliğin artırılması önemli katkılar sağlayacaktır.

Konfigürasyon yönetimi, ürünün ömür döngüsünde ürünü oluşturan parçaların üretim hatalarına karşı kalite ve performans iyileştirmesine yönelik olarak izlenmesi; düzeltilmesi için gereken mali, zaman ve iş gücü kaynaklarına önemli ölçüde azaltılmasına katkı sağlamaktadır.

2.10 Kaynak Yönetim Planı

Projede çalışan ekibin Helezonik Modelin her aşamasında olması, her bölümün teker teker gözlemlenmesi, ölçülendirilmesi ve test edilmeye gerek duyulduğundan dolayı Kaynak Yönetim Planı çok önemlidir. Kaynak Yönetim Planında yapılması gerekenler şunlardır;

- Planlamada temel hedeflerin açık anlaşılır bir şekilde belirlenmesi
- Risk yönetimi için temel hedeflerin belirlenmesi
- Planlamadan oyuncu ve müşteri değerlendirmelerine kadar olan bölümde(Planlama, Risk Yönetimi, Üretim, Oyuncu ve Müşteri Değerlendirmesi) uygun bütçenin çıkarılması
- Kabul edilmiş risklerin belirlenmesi
- Risk yönetim işlemleri, yönetim ve tekniklerinin tanımlanması
- Proje ile ilgili gerekli önceliklerin belirlenmesi

2.11 Eğitim Planı

Proje kapsamında Sinema Bilet Satış Otomasyonu kullanacak olan sinema firmalarında çalışan yetkili personele kullanıcı komutları hakkında eğitici bilgi verilecektir. Personele nasıl bilet satılacağını, sisteme nasıl film, salon numarası, seans saati, vb. işlemlerin nasıl yapılacağını bir seminer verilerek anlatılacaktır.

2.12 Test Planı

Yapılan program tüm Windows işletim sisteminde çalışabilmektedir. Yazılımı müşteri ile deneyerek programın düzgün çalışıp çalışmadığını test etmiş olacağız.

2.13 Bakım Planı

Sistem offline bir sistem olduğu için fazla bakım ihtiyacı duymaz. Nitekim gelişen elektronik sanayi ile yeni bazı güncellemeler gerekebilir. Bunun olması programın yeni sistemlere entegre edilmesini gerektirir. Bakım aşaması ile bu sorunlar ortadan kaldırılabilir.

Programımızda uygulanabilecek bakım türleri şöyledir;

- **Düzenleyici Bakım: Tespit** edilen hataların giderilmesi işlemidir. Kodlama hatalarının düzeltmek genelde az maliyetlidir. Tasarımdan kaynaklanan hataların giderilmesi ise bazı sistem bileşenlerinin baştan yazılmasını vb. gerektirebilir ve nispeten yüksek maliyetlidir.
- **Uyarlayıcı (Adaptif) Bakım: Yazılımın** yeni bir çalışma ortamına uyarlanmasıdır. Bu bir donanım platformu değişikliği olabileceği gibi (32-bit ten 64-bit e geçiş gibi) farklı bir işletim sistemine uyarlama şeklinde de olabilir(oyunun ve dolayısıyla yazılımın da Windows'tan Linux'a taşınması gibi).
- **İyileştirici Bakım: Sisteme** yeni işlev ve özelliklerin eklenmesi, performansın artırılması gibi bakım çalışmalarıdır.

3. Sistem Çözümleme

3.1 Mevcut Sistem İncelemesi

MEVCUT SİSTEM AKSAKLIKLARI:

- Sinema bileti satışında fazla sıranın olması
- Fazla sıranın olmasından dolayı personelin fazla yorulması
- Biletlerin yanlış kesilmesi
- Müşterinin istediği koltuğu seçememesi

3.2 Var olan ve Gereksenen Sistem

3.2.1 Örgüt Yapısı

- Müşteri istediği filmin ve seansı personelden talep eder
- Personel istenilen filmi ve seansı seçer
- Müşteri boş/dolu koltuklar arasından boş olanlardan seçim yapar
- Personel seçilen koltuk ve seanstan müşteriye bilet keser
- Personel yeni gelen filmi; seans, saat, salon numarası, vb. kategorilere göre kayıt yapar
- Gün sonunda satılan bilet sayısını Tam-Öğrenci olarak hesaplar ve yöneticiye çıktı olarak verir.



Yönetici



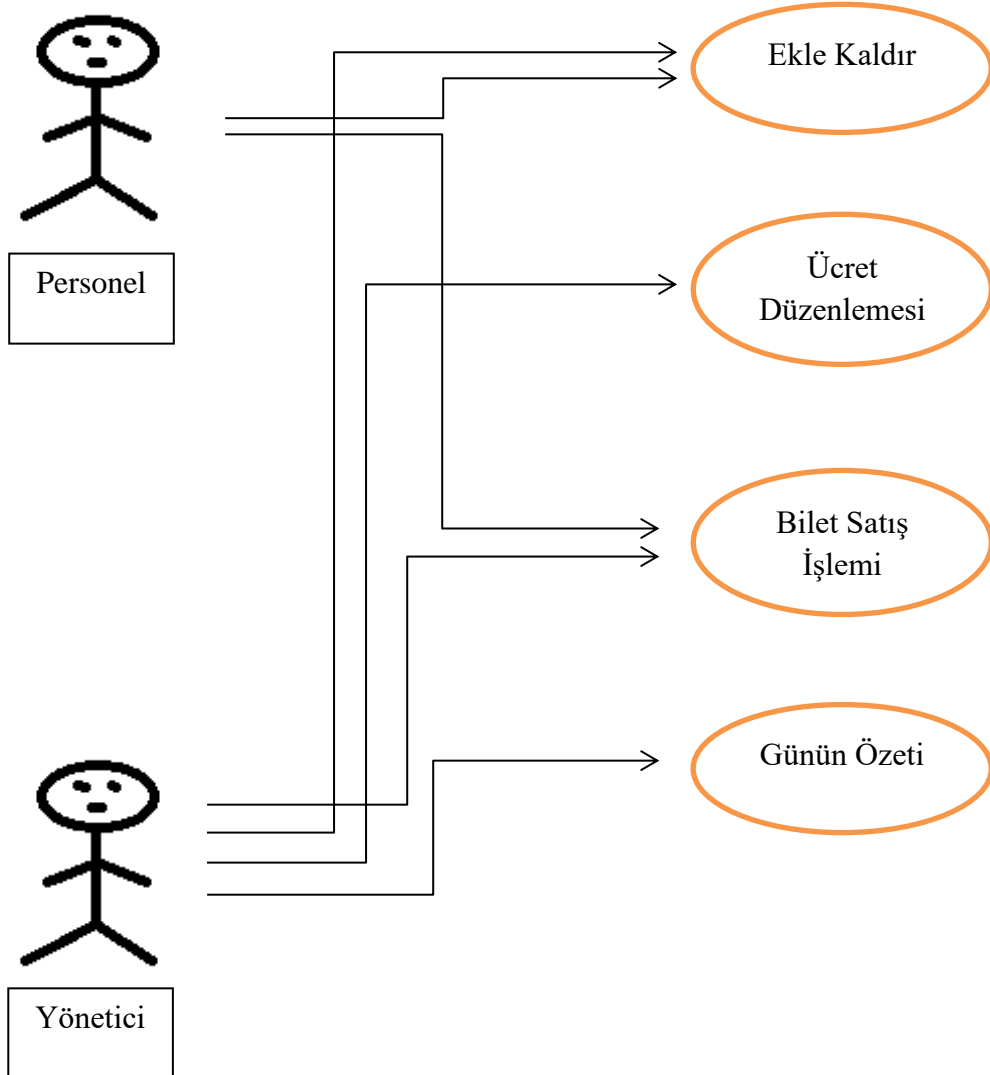
Personel



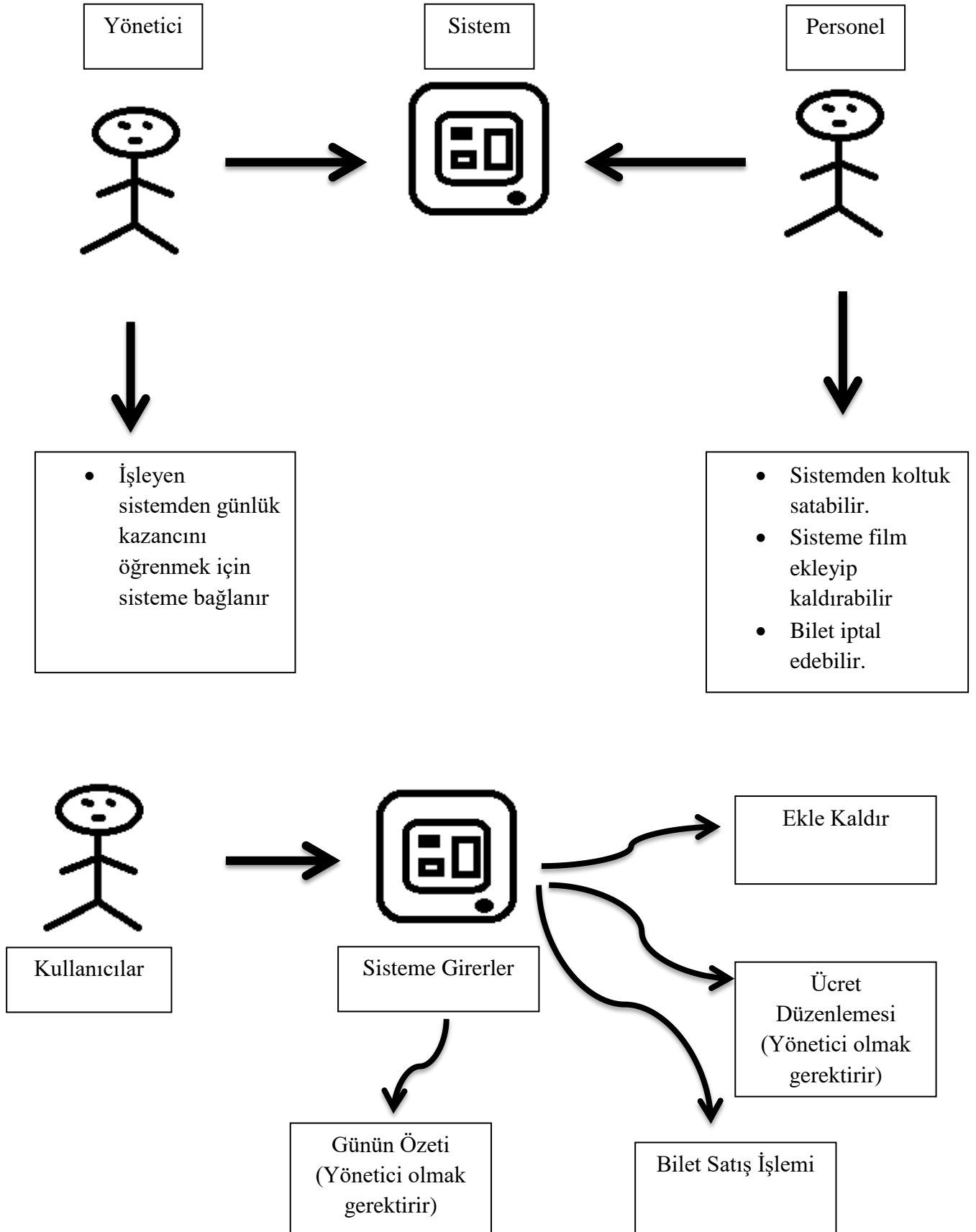
Müşteri

3.2.2 İşlevsel Model

- Yazılım, film ismi seçimi yaptıktan sonra başlayacak
- Seçilen filmden sonra yazılım bizden seans saatini seçmemizi isteyecek ve hemen ardından boş koltuk seçmemizi isteyecek
- Seçilen film, zaman ve seansa göre önceden personelin girmiş olduğu salon numarasın, göre yazılım çıktı hazırlayıp yazıcı yoluyla çıktıyı kağıda basacak.
- Gün bitiminde müdür için kaç tane bilet satıldığına dair çıktı çıkaracak.
- Yeni film gelirse veya önceki film vizyondan kalkarsa personel “Film Ekle/Kaldır” seçeneği ile filmleri düzenleyecek.



3.2.2.1 Use Case Diyagramı



Use Case Adı: Sistem

Aktör: Yönetici + Personel

Aktif olmayan Aktör: Müşteri

Olay: Kullanıcıların sisteme girebilmesi için bir parolaya ihtiyacı vardır

Olay Akışı: Kullanıcılar (Yönetici veya personel) parola kabul edildiği takdirde sistem arayüzü ile karşılaşır. Yönetici ile Personel için farklı arayüzler ortaya çıkar.

Use Case Adı: Ekle Kaldır

Aktör: Kullanıcılar

Aktif Olmayan Aktör: Müşteri

Olay: Vizyona yeni film gelmesi veya eski olan filmin vizyondan kaldırılmasına bağlı olarak sistemin düzenlenmesi gerekmektedir.

Olay Akışı:

- Ekleme işleminde bizden film ismi istenir.
- Film isminden sonra salon numarasını ve seans saatini ister.
- Vizyondaki film sisteme kaydedilir.
- Film kaldırma işleminde ise sadece filmin ismi yeterlidir

Use Case Adı: Bilet Satış İşlemi

Aktör: Kullanıcı(Personel) ve Müşteri

Aktif Olmayan Aktör: Kullanıcı (Yönetici)

Olay: Müşterinin isteğine göre Kullanıcının (Personel) bilet kesmesi

Olay Akışı:

- Personel müşteri biletinin ücretlendirme tipini seçer.
- Müşteri izlemek istediği filmin ismini söyler
- Personel istenilen filmi sistemden girer ve sistem seansları gösterir.
- Personel boş koltuğun olduğu seansı müşteriye gösterir ve müşteriden boş olan koltuklardan seçim yapması istenir.
- Yapılan seçime göre sistem film ismi, seans saati, salon numarası vb. isteklere göre müşteri için yazıcıdan bilet çıkartır.

Use Case: Ücret Düzenleme:

Aktör: Kullanıcı (Yönetici):

Aktif Olmayan Aktör: Kullanıcı(Personel) ve müşteri

Olay: Sinema biletinin ücreti

Olay Akışı:

- Yönetici Bilet ücretini istediği gibi ayarlayabilir
- Tam ve öğrenci olarak iki veya daha fazla seçenekle biletler farklı tiplerde fiyatlandırılabilir.

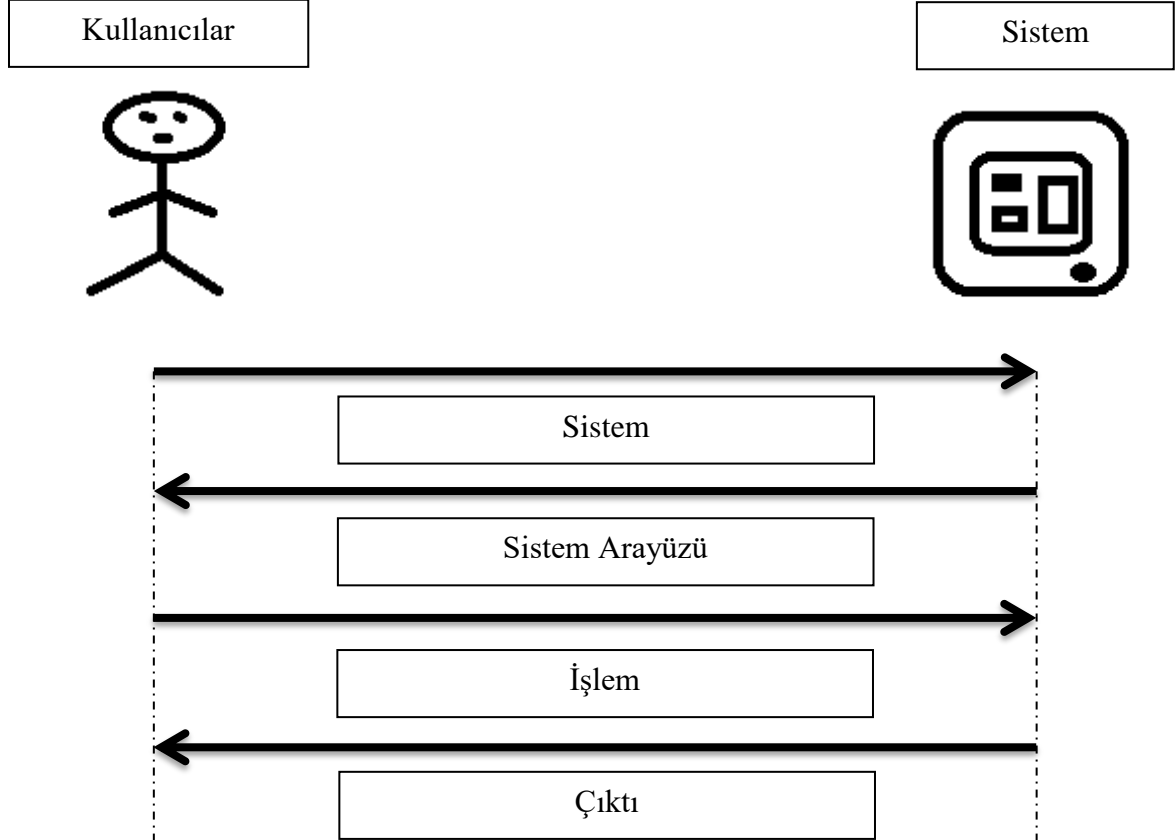
Use Case: Günün Özeti

Aktör: Kullanıcı(Yönetici)

Aktif Olmayan Aktör: Kullanıcı(Personel), müşteri

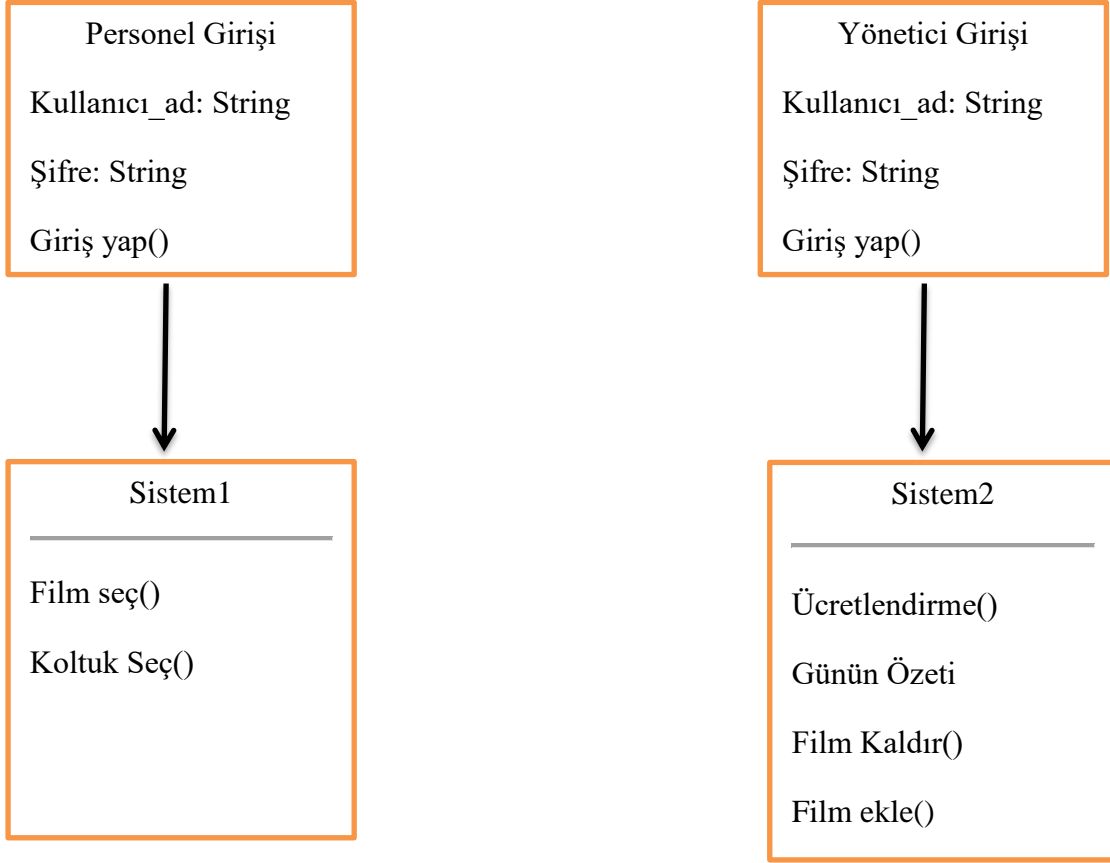
Olay: Günün bitiminde satılan bilet sayısı ve bilet başına düşen toplam gelirin çıktısı.

3.2.2.2 Durum Diyagramı



3.2.3 Bilgi Sistemleri/Nesneler

3.2.3.1 Sınıf Diyagramları



3.2.4 Veri Modeli



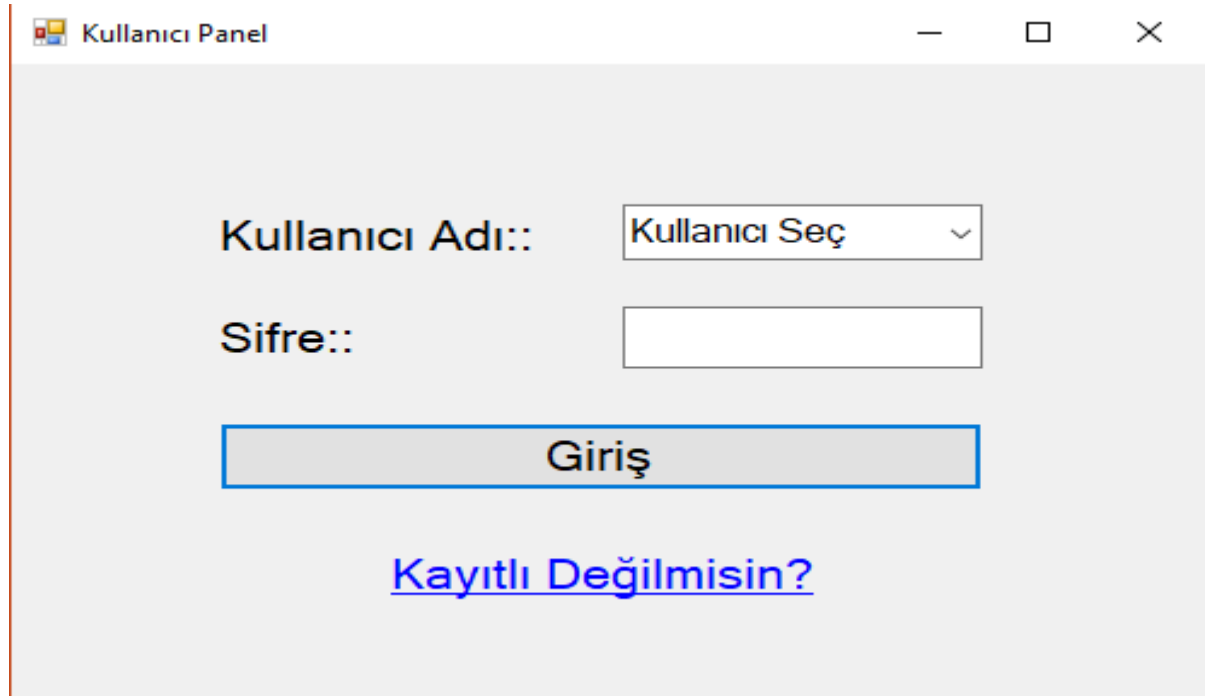
3.3 Arayüz (Modül) Gereklere

3.3.1 Yazılım Arayüzü

- Kullanıcılar (Yönetici, Personel) daha önce sisteme kayıt yaptırmadıysa bilgilerini giriş yapacak ve girilen bilgiler sisteme kayıt edilecek.
- Kullanıcılar sisteme giriş yaparak arayüze erişim sağlarlar.
- Yönetici ve personel arayüzleri farklıdır.
- Personel panelinde film seç, film ekle, film kaldır ve satılan bilet sayısı vardır.
- Yönetici panelinde ise ücretlendirme ve günün özeti vardır.
- Personel film seç e tıkladığında ise film seçme paneli ortaya çıkar ve aratılan filmde sonradakoltuk seçimi paneli ortaya çıkar.
- Personel film ekle butonuna tıkladıktan sonra film ekleme paneli ortaya çıkar.
- Film ekleme panelinde yeşil olan koltuklar boş kırmızı olan koltuklar ise doludur.

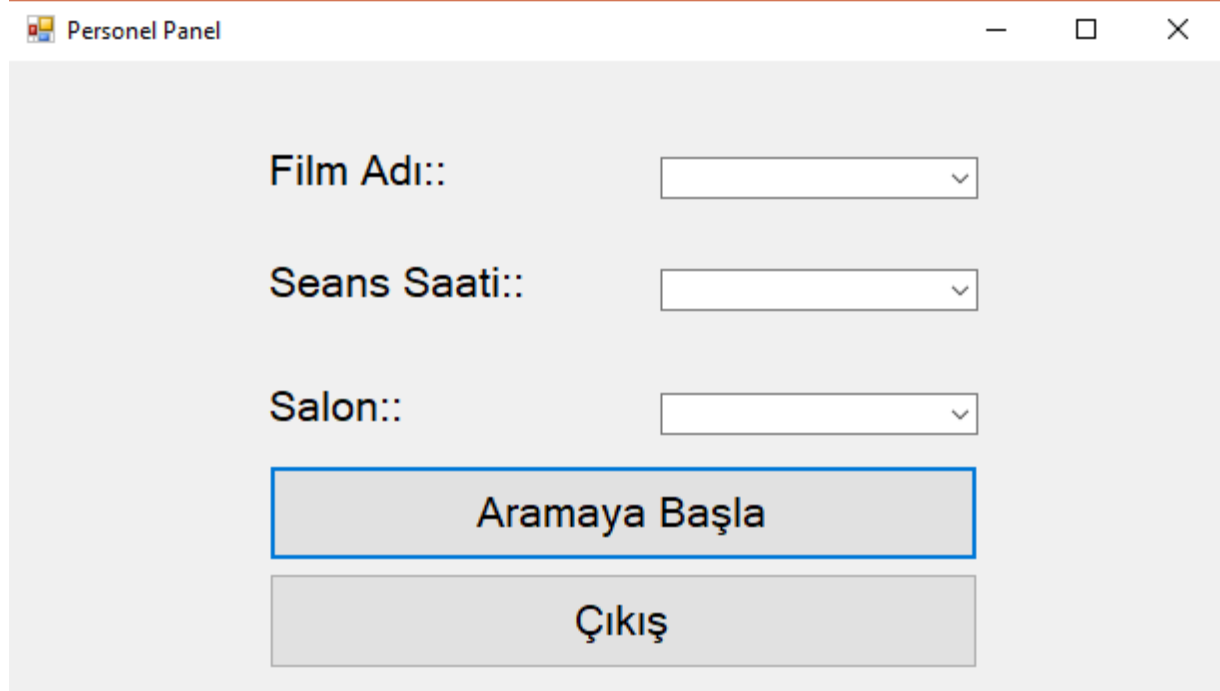
3.3.2 Kullanıcı Arayüzü

Kullanıcı Girişi:



- Kullanıcı girişi; yönetici ile personelin farklı panellere girmesini sağlar.

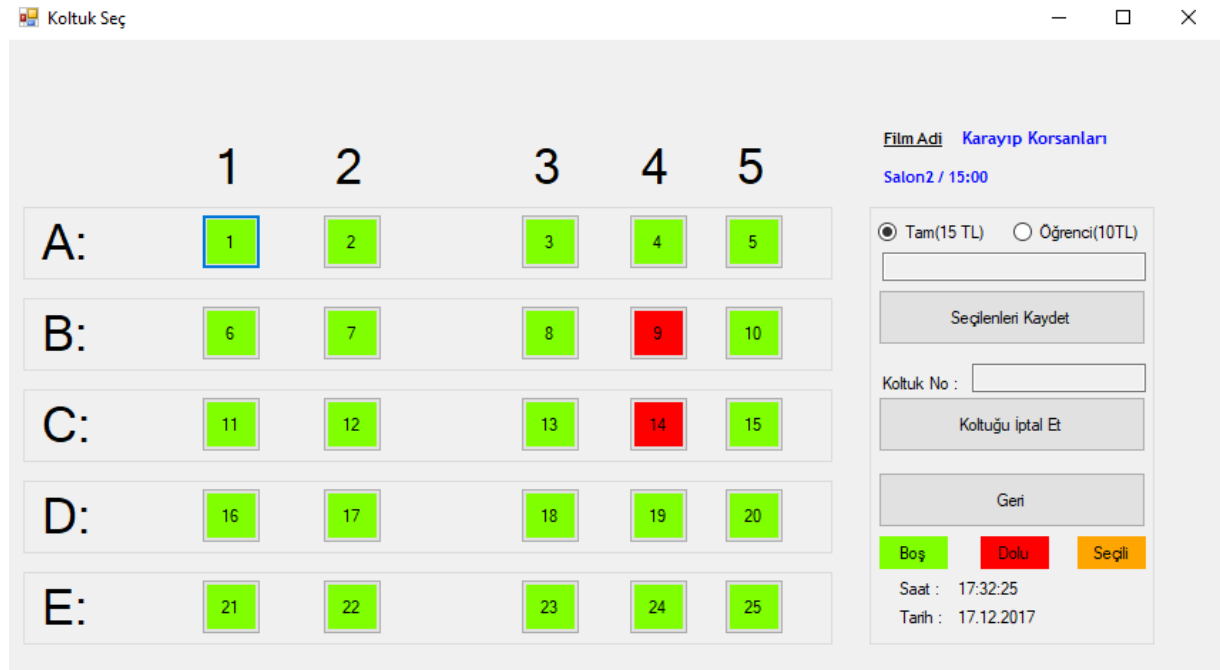
3.3.3 Personel Paneli



The screenshot shows a window titled "Personel Paneli". It contains three dropdown menus labeled "Film Adı::", "Seans Saati::", and "Salon::". Below these is a large button labeled "Aramaya Başla" and a smaller button labeled "Çıkış".

- Önceden kayıt edilmiş filmleri seçmemizi sağlar.
- Seçim yapılırken program bizden film ismini, seansı ve ücretlendirme seçmemizi ister.
- Yapılan seçimlere göre arama yapılır ve koltuk seçim paneli açılır.

Koltuk Seçim Paneli:



The screenshot shows a window titled "Koltuk Seç". It displays a grid of 25 seats arranged in 5 rows (A-E) and 5 columns (1-5). Seats are color-coded: green for empty, red for full, and orange for selected. In row A, seat 1 is selected. In row B, seat 9 is full. In row C, seat 14 is full. In row D, seat 19 is full. In row E, seat 24 is full. On the right side, there is a sidebar with the following information: "Film Adı: Karayıp Korsanları", "Salon2 / 15:00", "Tık (15 TL) / Öğrenci (10 TL)", "Seçilenleri Kaydet", "Koltuk No:", "Koltuđu İptal Et", "Geri", "Boş", "Dolu", "Seçili", "Saat: 17:32:25", and "Tarih: 17.12.2017".

- Bu panelde yeşil olan koltuklar boş olduğunu temsil ederken kırmızılar ise dolu olduğunu temsil eder.
- Bu panel müşteriye gösterilerek müşterinin koltuk seçmesi istenilir.

- Seçilen koltuklar onaylandıktan sonra yazıcı yardımı ile çıktı alınır.
- Çıktıda Koltuk numarası, filmin ismi, seans tarih ve saati, salon numarası yazılmaktadır.

3.3.3.1 Film Ekle me Arayüzü

Admin Film Paneli

Güncelle

Sil

Kayıt

Geri

	film_no	film_ad	film_yönetmen	film_tur
▶	2	Thor	dsad	Macera
	3	Karayıp Korsanlar	fadsfdsa	dsafdsaf
	15	Ayla	Ahmet	fsafdsaf
	16	Çalgı Çengi	sdasdsad	dasdsadsadsad
	17	Düğün demek 2	mog	game
	18	ddsadda	sdadsa	123321sd
	20	sasas	sasa	sasas
*				

Film Ekle

Film İsmi:

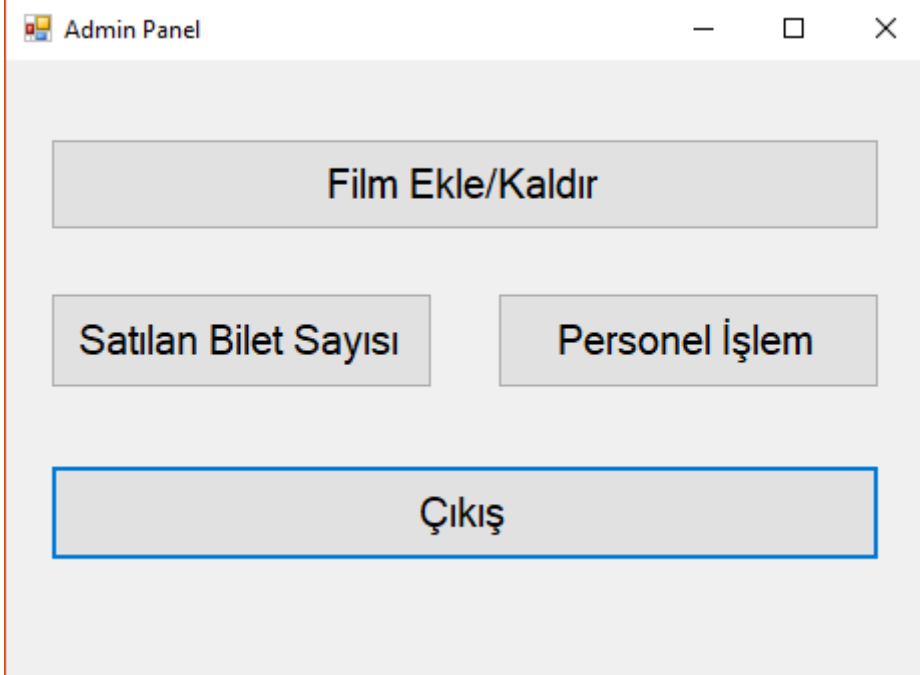
Filmin Türü:

Yönetmen Adı:

Kayıt

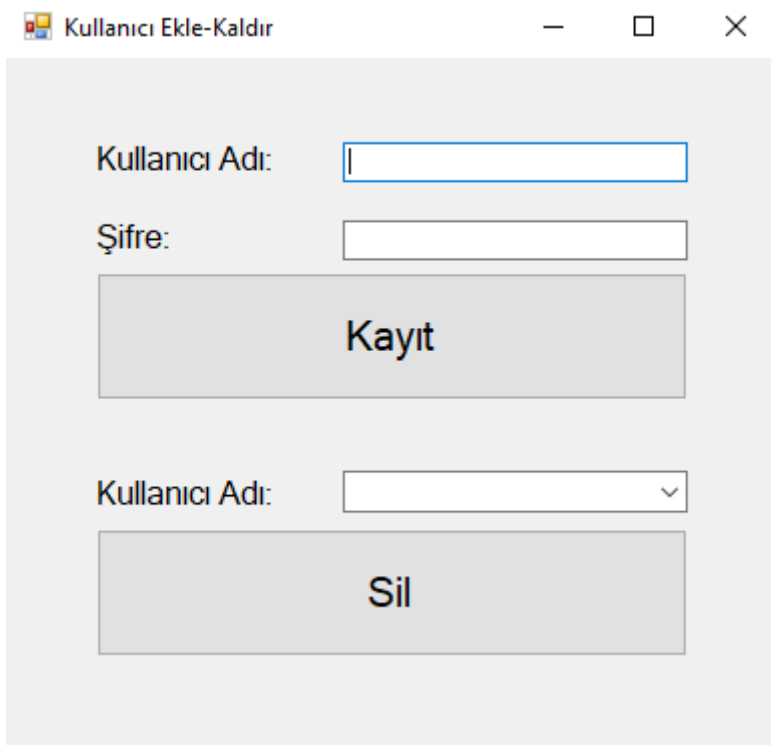
- Bu panelde filmin verileri sisteme eklenir.
- Eklenen filmler daha sonradan kullanılır.

3.3.4 Yönetici Arayüzü



- Yönetici arayüzünde; ücretlendirme ve günün özeti vardır.
- Günün özeti ise gün içinde yapılan satışların kaydı tutulur ve o kayıt yazıcı ile çıktı olarak verilir.

3.3.5 Personel İşlem



Kullanıcı Ekle-Kaldır

Kullanıcı Adı:

Şifre:

Kayıt

Kullanıcı Adı:

Sil

3.4 Belgeleme Gerekleri

3.4.1 Geliştirme Sürecinin Belgelendirmesi

Geliştirme süreci belirtilmiş olan kriterler doğrultusunda gerçekleştirilmiştir. Bu geliştirme süreçleri (V Süreç Modeli, Helezonik Model, Evrensel Geliştirme Süreç Modeli, Araştırma Tabanlı Süreç Modeli) arasından Helezonik Model kullanılmıştır.

3.4.2 Eğitim Belgeleri

Yazılımın kurulacağı sinema firması içerisinde teknik olarak bakım ve yazılım olarak, kullanıcılar için ayrı olarak doküman yazılmamıştır.

3.4.3 Kullanıcı El Kitapları

Sistem kullanıcılar için kolay bir kullanım sunduğu için kullanıcı el kitabı hazırlanmamıştır. Ama sistemin nasıl kullanılacağını seminer verilerek öğretilecektir.

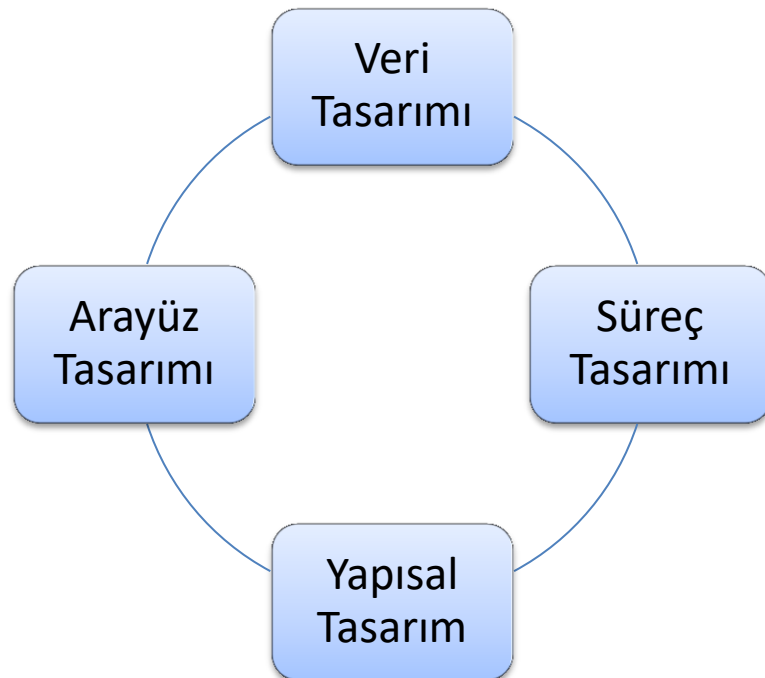
4. Sistem Tasarımı

4.1 Genel Tasarım Bilgileri



4.1.1 Genel Sistem Tasarımı

Temel hedef, yazılım yapısına ait diyagramları oluşturmaktır. Veri akış diyagramlarından yola çıkarak sistemdeki işlerin işleyişinin tanımlanmasıdır.

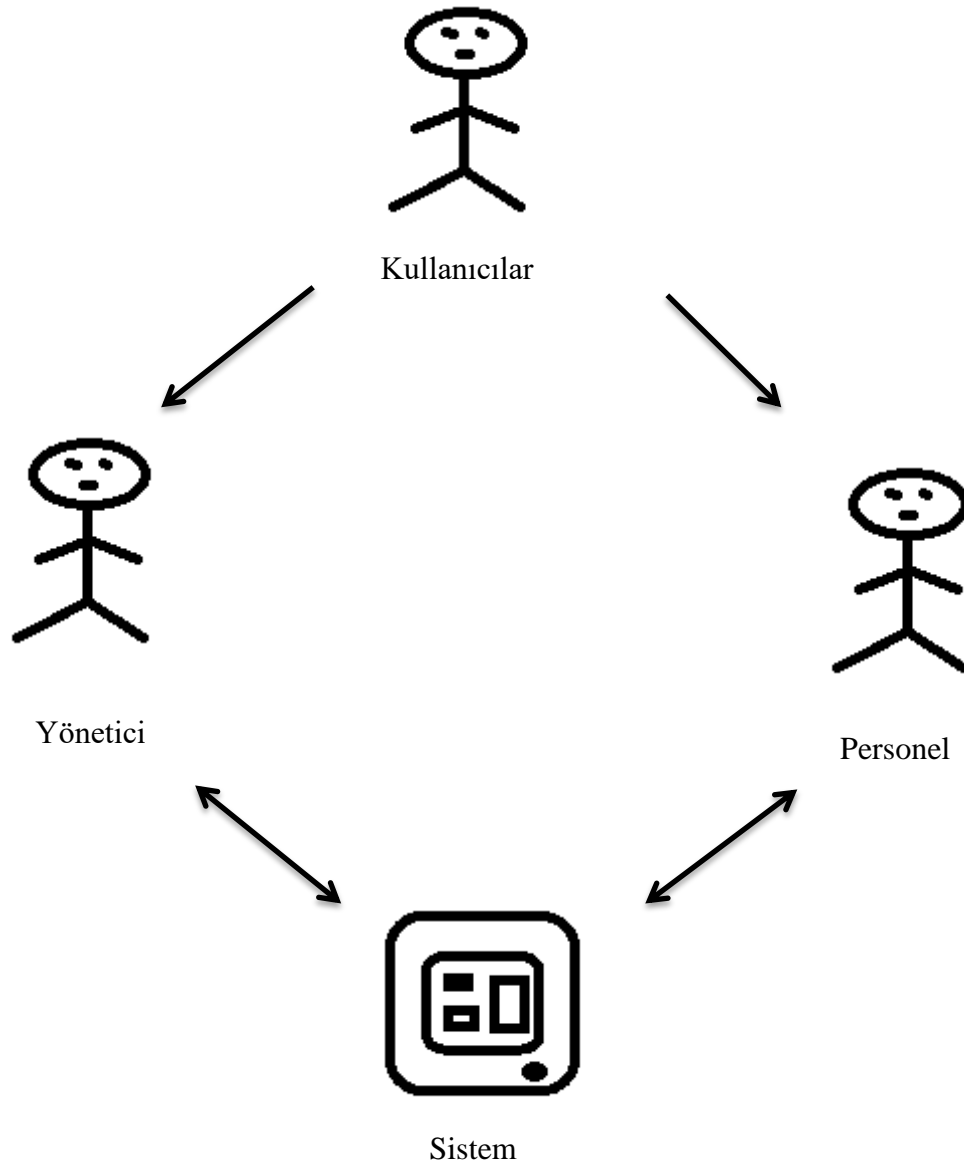


4.1.2 Varsayımlar ve Kısaltmalar

- Proje bitmesi gereken zamanda bitecektir
 - Proje sonuna kadar bir ekip çalışması olacaktır
 - Projenin her aşamasında proje yöneticiyle toplantılar yapılacaktır
 - Proje sonuna kadar eklenen her yeni bilgiden tüm ekibin haberdar olması gerekmektedir
 - Projenin kısa zamanda ve düşük maliyette bitirilmesi amaçlanmaktadır
-
- **Sistem;**
 - Personeller sisteme girebilmesi için kayıtlı olmaları gerekmektedir.
 - Sistemi sadece yönetici ve personeller kullanabilir.
 - Personeller film ekleyip kaldırabilir ve bilet satışı yapabilirler.
 - Yöneticiler ücretlendirmeleri ayarlayabilir ve günün özetini görebilirler.

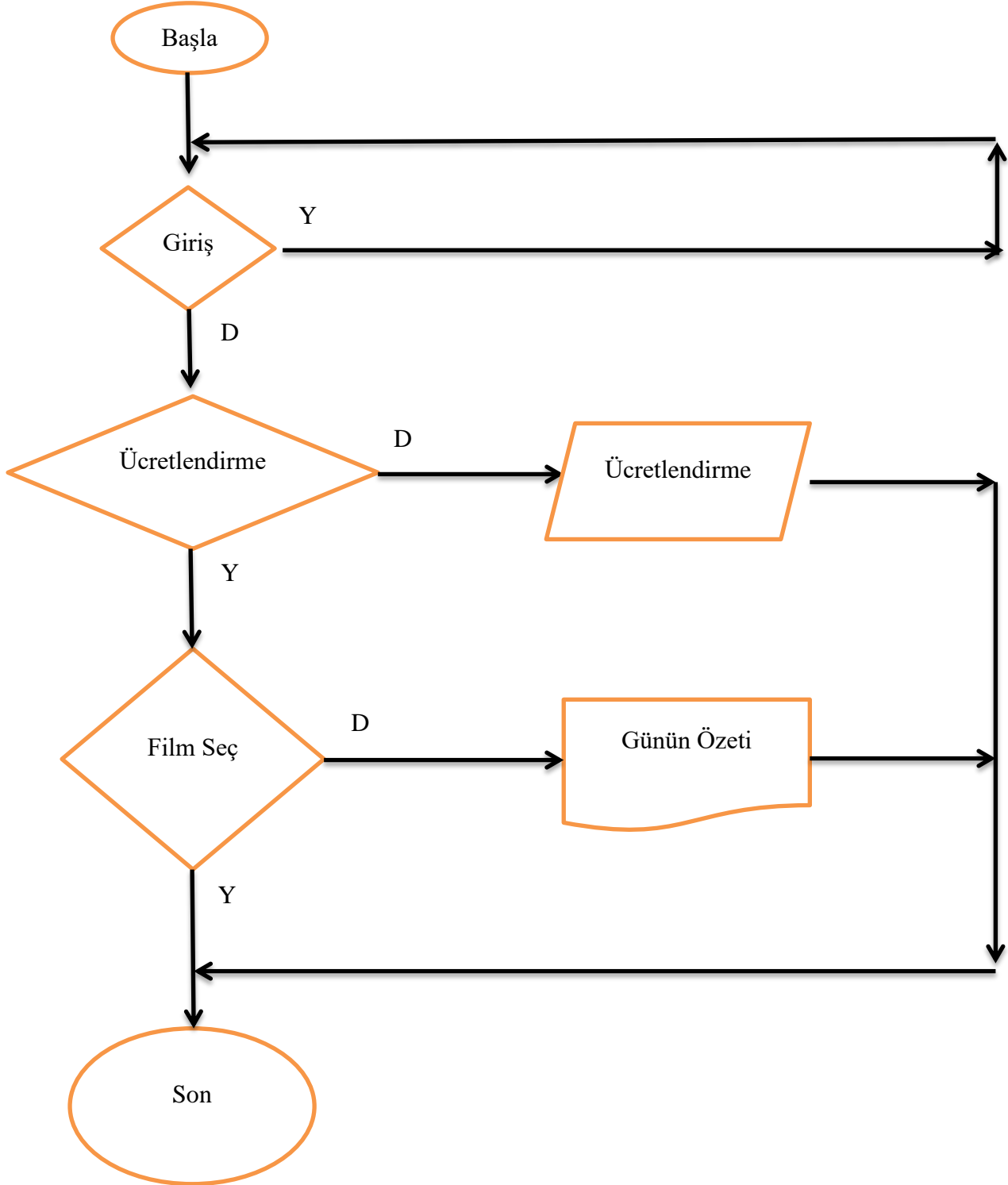
Affinity Diagram		
İşlemler	Yönetici	Personel
Arayüzü Görme	+	+
Giriş	+	+
Kayıt Olma	-	+
Film Seçme	-	+
Film Ekle/Kaldır	-	+
Ücretlendirme	+	-
Günün Özeti	+	-

4.1.3 Sistem Mimarisi

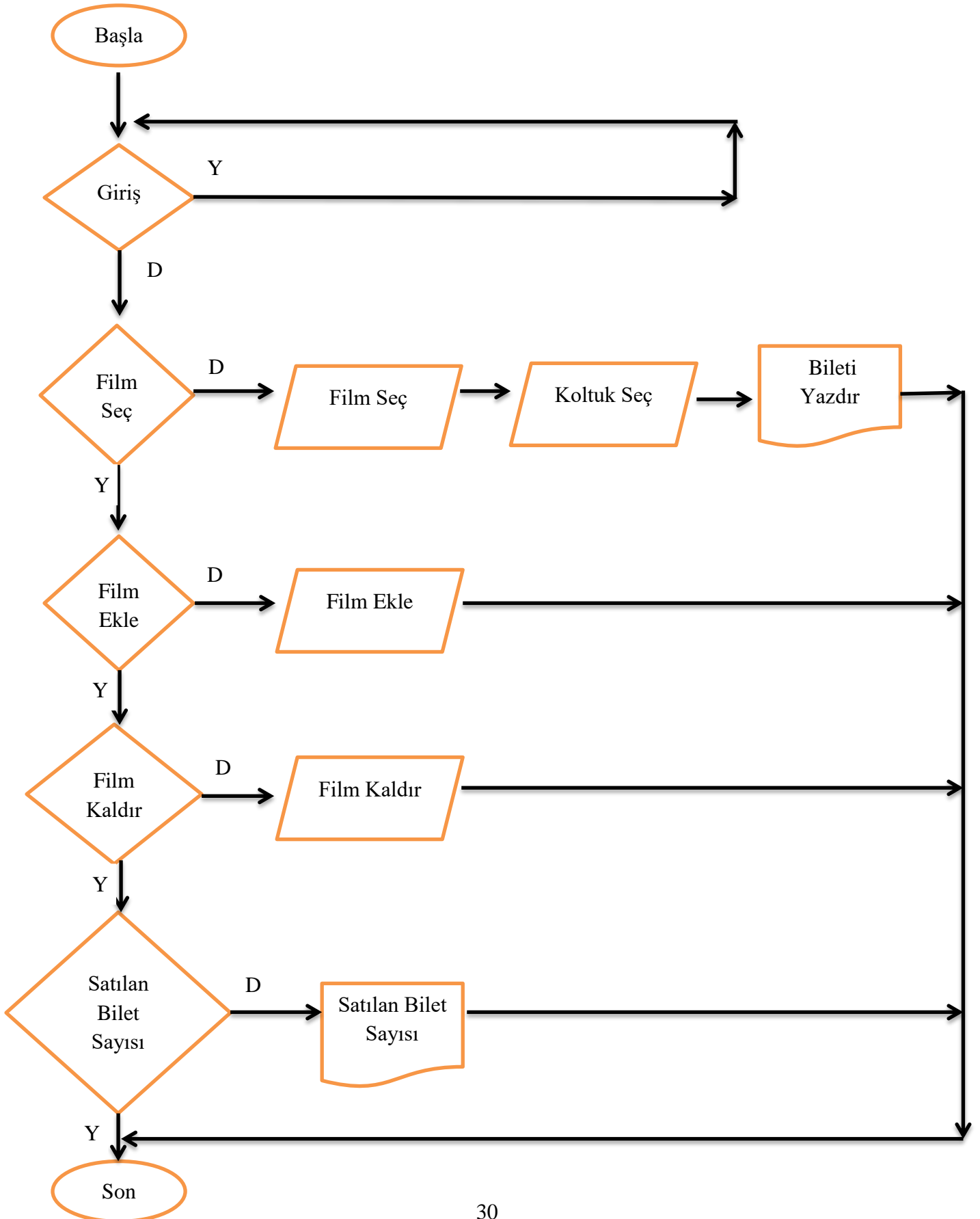


4.1.3.1 Akış Diyagramları

4.1.3.1.1 Yönetici



4.1.3.1.2 Personel



4.1.4 Dış Arabirimler

4.1.4.1 Kullanıcı Arabirimleri

4.1.4.1.1 Personel Arabirimi

Personellerin kullanabildiği arabirimlerdir. Bu arabirimler:

- Film Ekle
- Film Kaldır
- Film Seç
- Satılan bilet sayısı



4.1.4.1.2 Yönetici Arabirimi

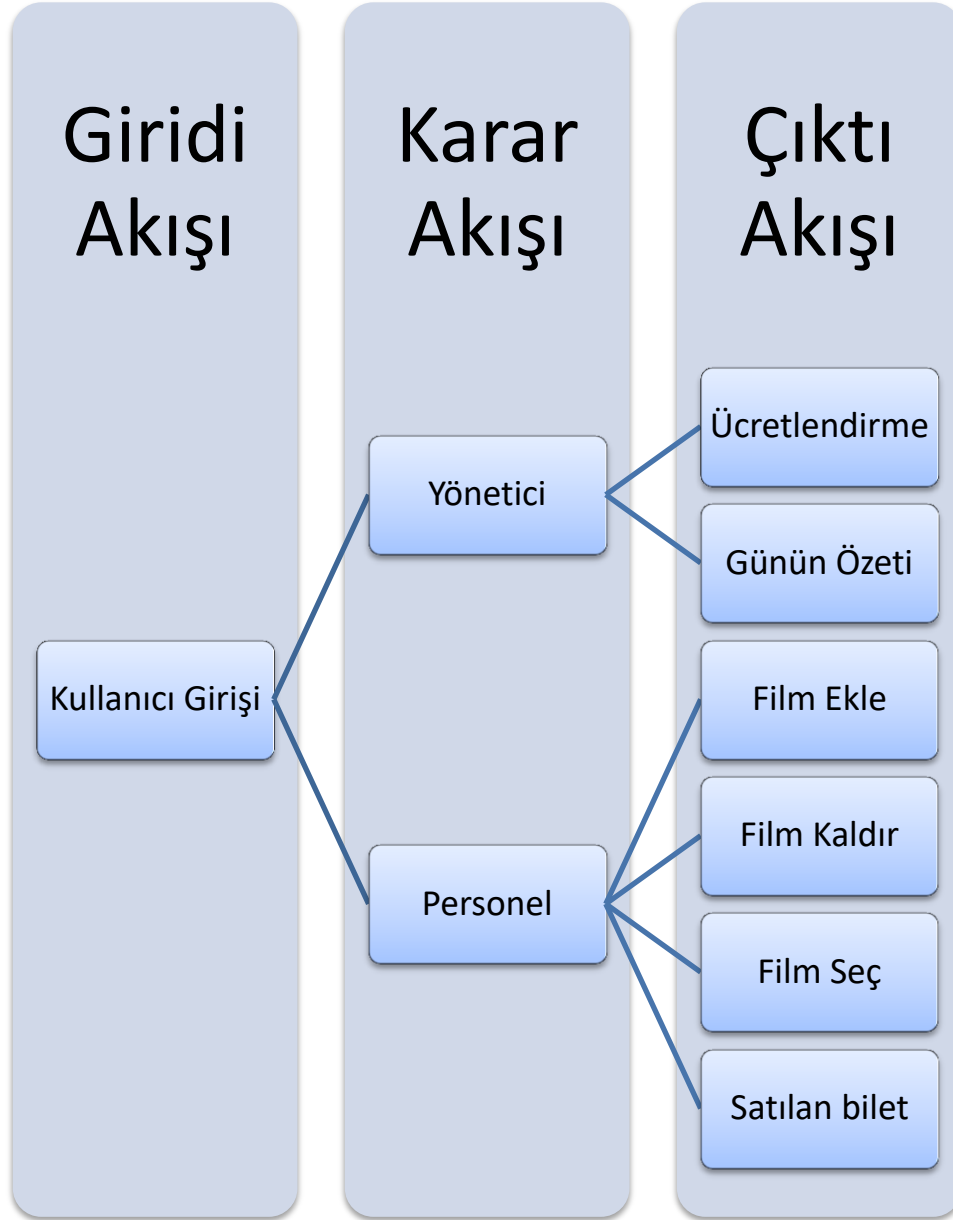
Yöneticilerin kullanabildikleri arabirimler:

- Ücretlendirme
- Günün Özeti

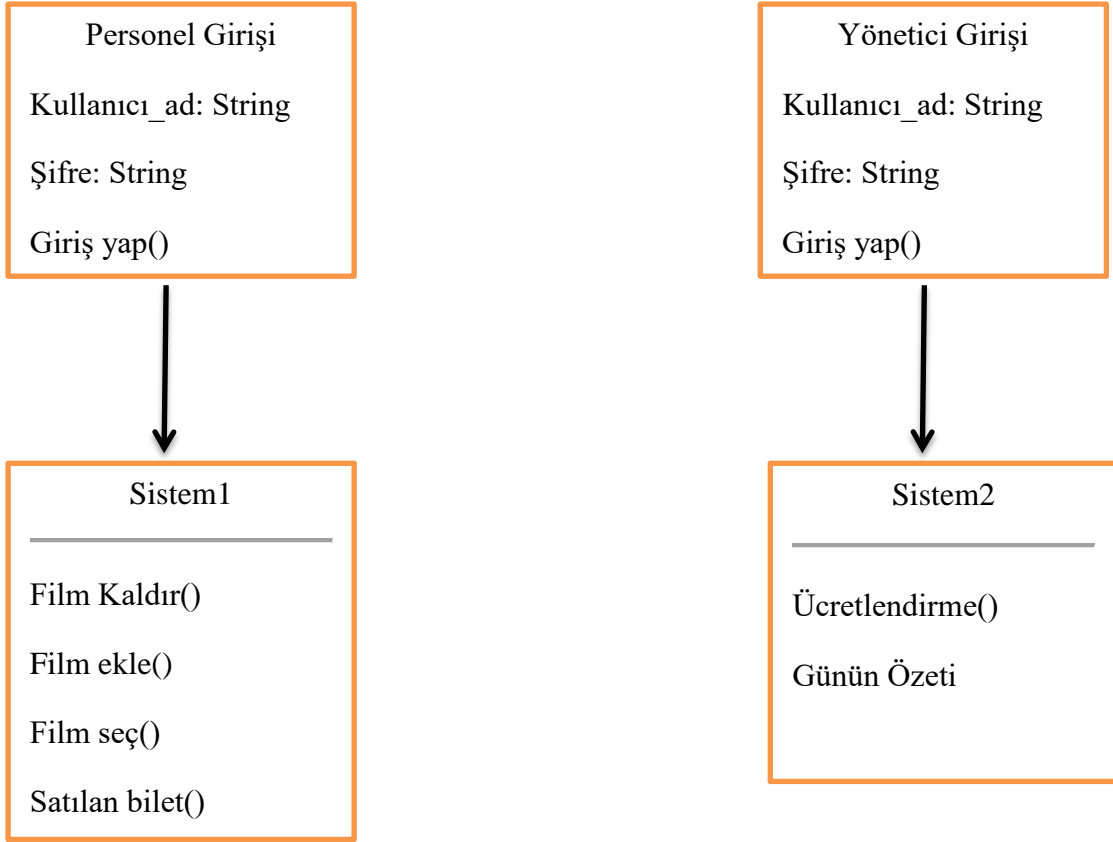


4.1.4.2 Veri Arabirimleri

Sistem veri akışını üç bölüme ayırmak mümkündür. Girdi Akışı, Karar Akışı, Çıktı Akışı



4.1.5 Veri Modeli



4.1.6 Testler

Proje yapımı boyunca oluşabilecek her türlü sorun test edilecek ve aynı zamanda yazılımın kendi sistem ve donanım hatalarına dair raporları da test aşamasına etki edecektir.

4.1.7 Performans

Yazılımın belirlenen süreç modeline(Helezonik Model) göre yönetilmesi, gerekli dokümantasyonun hazırlanmasında ve plana bağlı yazılımın gerçekleştirilmesi performans değerlerine öncülük etmektedir. Yapılan uygulamaların performansının daha iyi olması için gerekli bilgi test aşamasında detaylı bir şekilde aktarılmıştır.

4.2 Veri Tasarımı

Veri Tasarımını: Süreç Tasarımı ve Arayüz Tasarımından önce yapılması gereken ilk adımdır. Bilgi saklama ve soyutlama veri tasarımı için önemli kavramlardır.

Projemizde gerçekleştirme aşamasında veri tasarımı yaparken dikkat edilecek hususlar:

- Farklı veri yapıları değerlendirilmiştir.
- Bütün veri yapıları ve Üzerinde yapılacak işlemler belirlenmiştir.
- En çok kullanılacak olan veri yapıları için veri sözlüğü oluşturulmuştur.

4.2.1 Tablo Tanımları

Personel Girişi: Veri tabanında kullanıcıların sisteme erişebilmesi için kullanıcı adı ve şifre gibi bilgilerin bulunması gerekir.

Yönetici Girişi: Veri tabanında kullanıcıların sisteme erişebilmesi için kullanıcı adı ve şifre gibi bilgilerin bulunması gerekir.

Satılan Bilet Sayısı: Satılan bilet sayısı veri tabanında tutulacaktır.

Filmler: Eklenen filmler veri tabanında tutulacaktır.

4.2.2 Veri Tanımları

4.2.2.1 Personel/Yönetici Girişi

- Kullanıcı Adı

İşlev: Kullanıcının Sisteme giriş yapması için kullanılan karakter topluluğu.

Kullanım yeri: Sisteme giriş arayüzü

- Şifre

İşlev: Kullanıcının Sisteme giriş yapması için kullanılan karakter topluluğu.

Kullanım yeri: Sisteme giriş arayüzü

4.3 Süreç Tasarımı

4.3.1 Genel Tasarım

Yazılım geliştirme süreçleri sürekli yaşayan ve iyileştirilen süreçler olmak zorundadırlar. Dolayısı ile, bir süreç yönetim mekanizması kurulması zorunludur. Süreç yönetimi, süreç tanımlama, organizasyonel eğitim, süreç iyileştirme süreçleri şeklinde organize edilebilir.

- **Süreç Tanımlama Süreci:** Süreç Tanımlama Sürecinin amacı, iş süreç için tutarlı, tekrarlanabilir, performansını destekleyecek süreç kütüphanesi oluşturacaktır. Bu süreç için, süreç tanımlama formu, süreç iş çıktıları hazırlama formu gibi şablonlar tanımlanmalıdır.
- **Organizasyonel Eğitim:** Çalışanların kendilerine verilen rolleri verimli ve etkin bir şekilde gerçekleştirmelerini sağlamaktır.

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



4.3.2 Modüller

4.3.2.1 Yönetici Modülü

4.3.2.1.1 İşlev

Yönetici ücretlendirmeleri ayarlar ve Günün özetini çıktı olarak alır.

4.3.2.2 Personel Modülü

4.3.2.2.1 İşlev

Personelin Film ekleyip kaldırmasını, Film seçerek bilet satmasını ve satılan bilet sayısını görebilmesi için yapılmış bir modüldür.

4.3.2.3 Satılan Bilet Sayısı Modülü

4.3.2.3.1 İşlev

Sayaç yardımıyla gün içinde satılan bilet sayısını ve ücretini yazıcı yardımıyla çıktı haline getirir ve veri yapısı ile kayıt altına alınır. Böylece daha sonra o veriye erişim sağlanabilir.

4.3.2.4 Ücretlendirme Modülü

4.3.2.4.1 İşlev

Filmin ücretlendirilmesinin yapıldığı modüldür. Bu modüller ücretler tanımlanır ve yapılan satışlar bu ücretlendirme ile yapılır.

4.3.2.5 Film Ekleme Modülü

4.3.2.5.1 İşlev

Personelin Filmin ismine, Seansa, Salona, Filmin türüne ve yönetmene göre filmi eklemesi sağlanır. Eklenen film veri tabanı ile sisteme kaydedilir

4.3.2.6 Film Kaldırma Modülü

4.3.2.6.1 İşlev

Film kaldır modülü vizyondan kaldırılan film seçilerek filmi veri tabanından siler.

4.3.2.7 Film Seçim modülü

4.3.2.7.1 İşlev

Veri tabanından film ismi ve seansa göre arama yapılır. Eşleşen kriterlerde uygun film varsa modül bizi koltuk seçim modülüne götürür.

4.3.2.8 Koltuk Seçim Modülü

4.3.2.8.1 İşlev

Veri tabanında önceden kaydedilen yerlerin boş olup olmadığını kontrol eden modüldür. Boş olan yerleri yeşil ile gösterir iken dolu olan yerleri kırmızı ile gösterir. Kırmızı olan yerlere arayüzden tıklanma yapılamaz.

4.3.2.9 Satılan Bilet Sayısı

4.3.2.9.1 İşlev

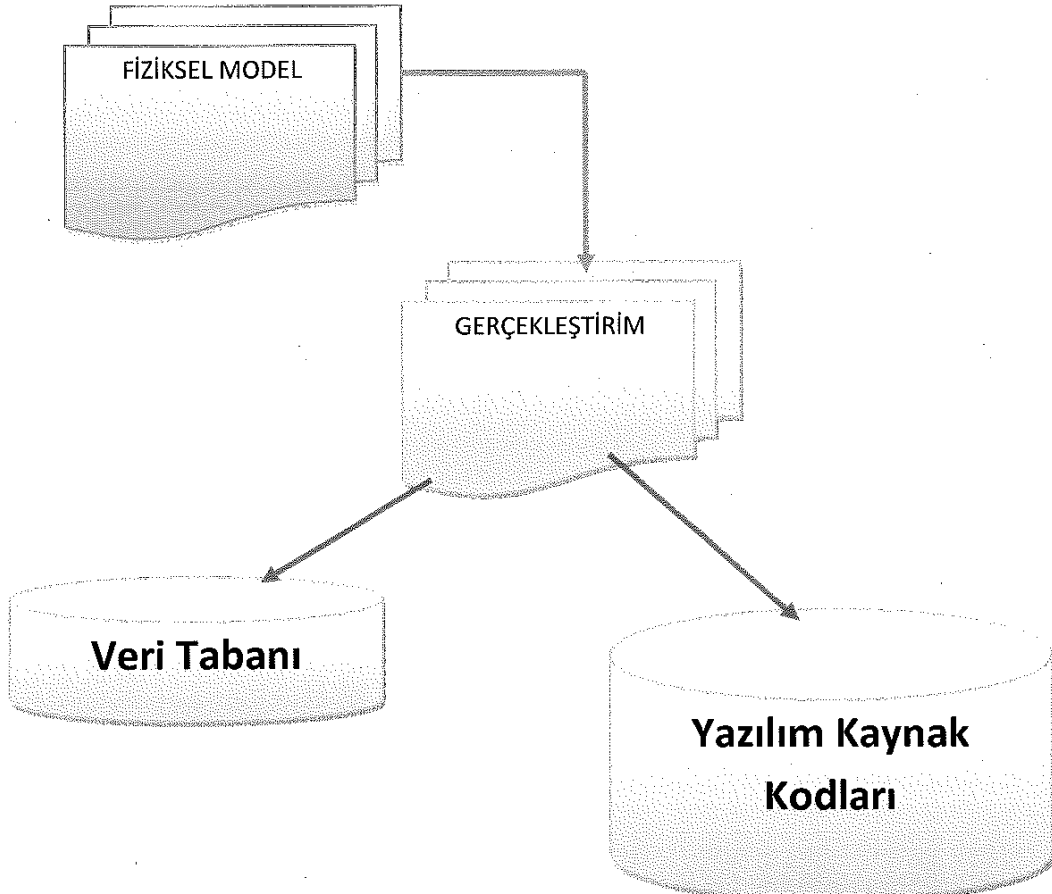
Satılan biletler veri tabanına kaydedilerek daha sonra ne kadar satış yapıldığını öğrenilmesi sağlanır.

5. Sistem Gerçekleştirimi

5.1 Giriş

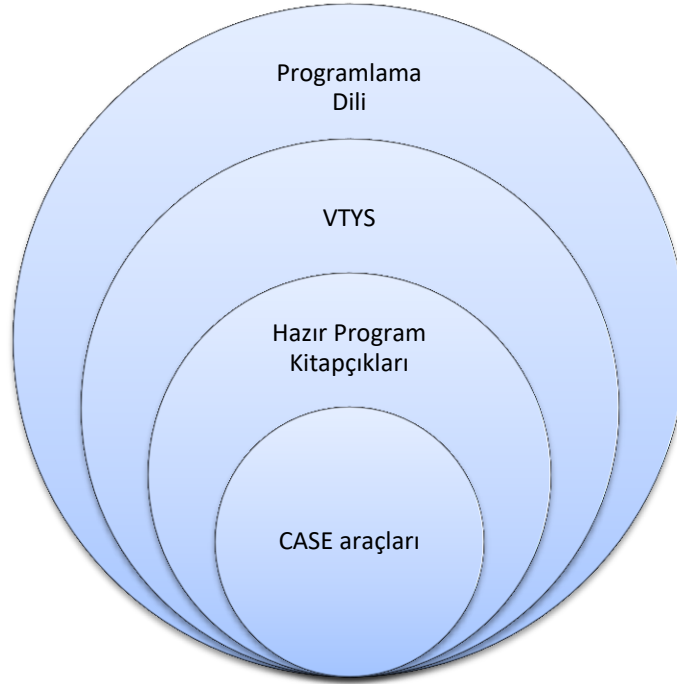
Gerçekleştirim çalışması, tasarım sonucu üretilen süreç ve veri tabanının fiziksel yapısını içeren fiziksel modelin bilgisayar ortamında çalışan yazılım biçimine dönüştürülmesi çalışmalarını içerir. Yazılımın geliştirilmesi için her şeyden önce belirli bir yazılım geliştirme ortamının seçilmesi gerekmektedir.

Söz konusu ortam, kullanılacak programlama dili ve yazılım geliştirme araçlarını içerir. Söz konusu ortamda belirli bir standartta geliştirilen programlar, gözden geçirilir, sınanır ve uygulamaya hazır hale getirilir. Üretilen kaynak kodların belirlenecek bir standartta üretilmesi yazılımın daha sonraki aşamalardaki bakımı açısından çok önemlidir. Tersı durumda kaynak kodların okunabilirliğe, düzeltilebilirliğe zorlanır ve yazılımın işletimi süresince ortaya çıkabilecek sorunlar kolayca çözülemez. Aşağıda genel hatlarıyla gerçekleştirim süreci gösterilmektedir;



5.2 Yazılım Geliştirme Ortamları

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

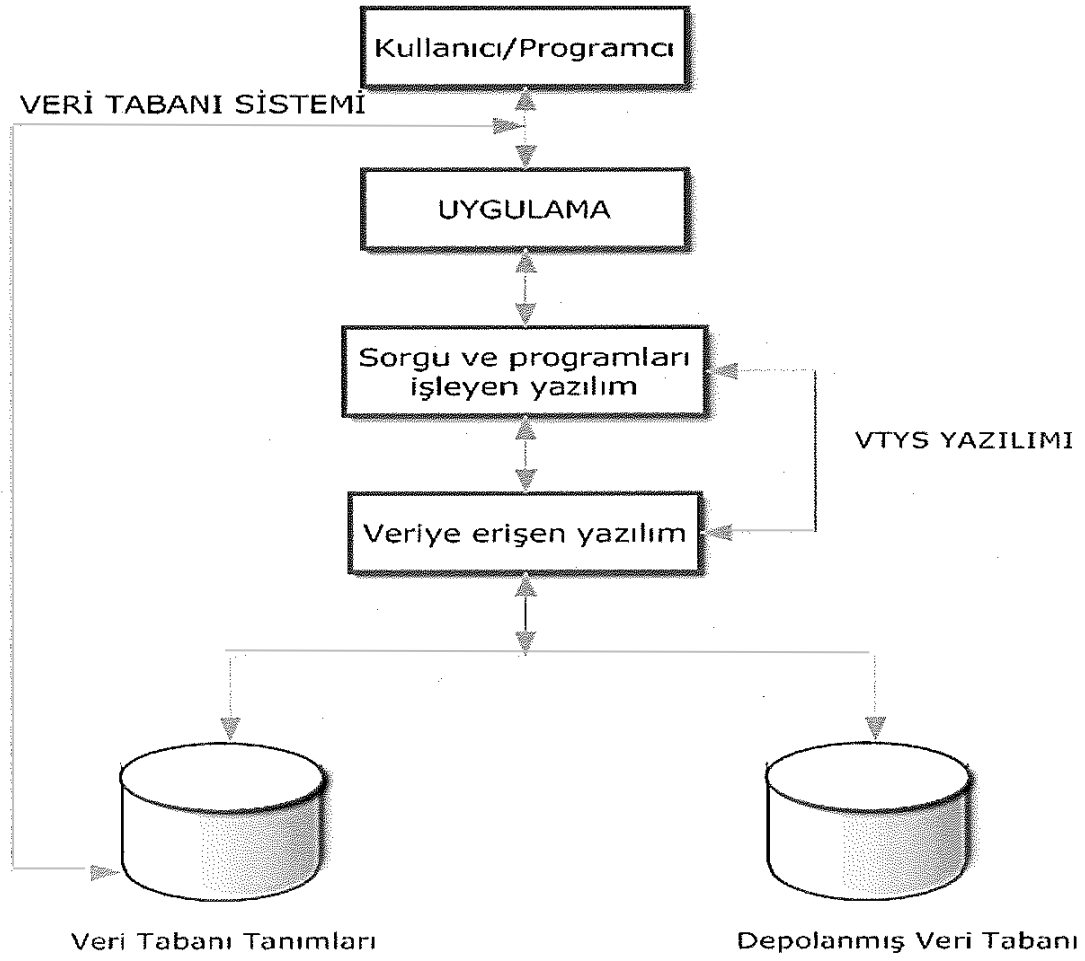


Bileşenlerinden oluşur. Günümüzde söz konusu bileşenler oldukça farklılık ve çeşitlilik göstermekte ve teknolojinin değişimine uygun olarak gelişmektedir.

5.2.1 Veri Tabanı Yönetim Sistemleri

Birbiri ile ilişkili veriler topluluğu veri tabanı olarak tanımlanmaktadır. Veri tabanı herhangi bir boyutta ya da karmaşıklıkta olabilir. Kişisel telefon rehberinizdeki adres bilgileri bir veri tabanı örneği oluşturduğu gibi maliye bakanlığı bünyesinde saklanan ve vergi ödemesi gereken tüm kişilerin bilgilerinin saklandığı uygulama da bir veri tabanı örneği teşkil eder. Veri tabanını oluşturan veriler birbirleriyle ilişkili verilerdir. Bir veri tabanında, veriler arası ilişki ile veri değerleri bulunur.

Kullanıcıların veri tabanındaki verileri soruşturmasını, veri tabanına yeni veriler eklemesini, var olan verilerde değişiklik yapmasını sağlayan yazılım Veri Tabanı Yönetim Sistemi(VTYS) olarak tanımlanır. VTYS, genel amaçlı bir yazılımdır.



5.2.1.1 VTYS Kullanımının Ek Yararları

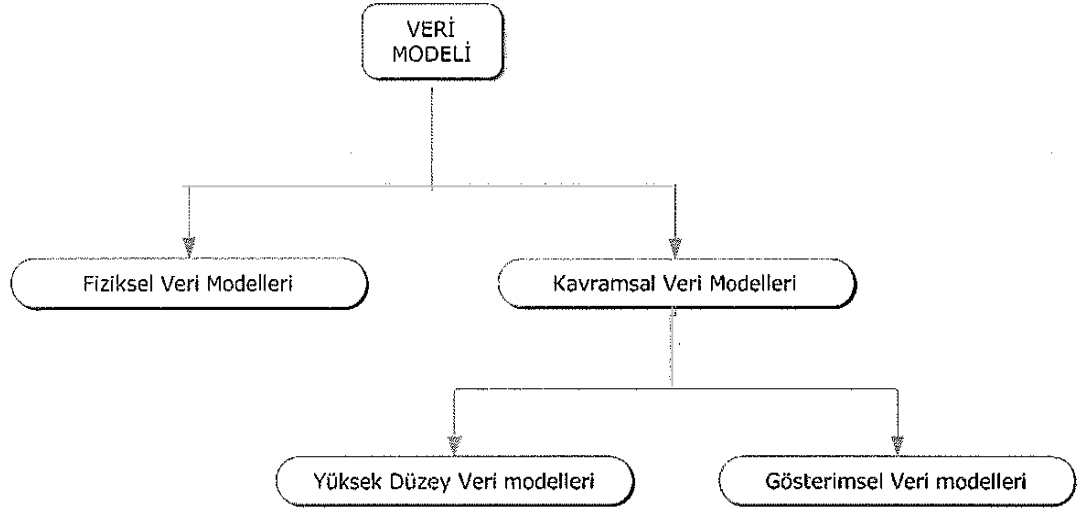
- Esneklik
- Genişleme potansiyeli yüksektir
- Uygulama geliştirme zamanı azalması
- Güncel bilgilerin tüm kullanıcılara zamanında ulaşması
- Ölçümde ekonomi
- Verilere hızlı erişim sağlar
- Verilerin yedeklenmesinde kolaylık sağlar
- İşletme ortamındaki verilerin tekrarının önlenmesi verilerin merkezi denetimini ve tutarlılığını sağlar
- Fiziksel yapı ve erişim yöntemi karmaşıklıklarının her kullanıcıya yalnız ilgilendiği verilerin kolay anlaşılır yapılarda sunulmasını sağlar
- Uygulama yazılımı geliştirmenin kolaylaştırılmasını sağlar
- Var olan verilerde değişiklik yapılmasını sağlar

Günümüzde VTYS yazılımları oldukça gelişmiş, görsel programlama platformları ve web tabanlı geliştirme platformları ile bütünleşik çalışma olanakları sağlamıştır. Artık geleneksel

kütük düzenleme ve erişim yöntemlerinin doğrudan programlanarak bilgi sistemi uygulaması geliştirilmektedir. VTYS yazılımları, kişisel bilgisayar düzeyine indirgenmiştir.

5.2.1.2 Veri Modelleri

Veri modelleri, veri tabanının yapısını anlamak için sağladıkları kavram tipine göre sınıflandırılır.



5.2.1.3 Şemalar

Herhangi bir veri modelinde veri tabanının tanımlanması ile kendisini ayırmak önemlidir. Veri tabanının tanımlamaları veri tabanı şeması veya tema-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.

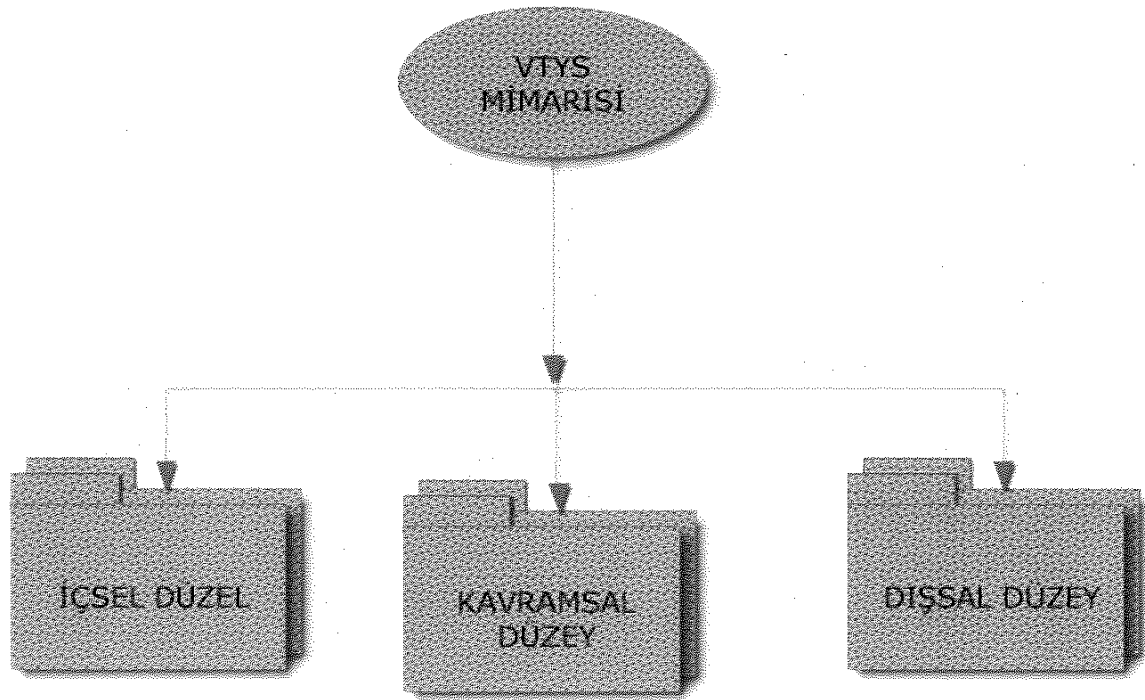
Bir veri tabanındaki gerçek veri sıkça değişebilir. Herhangi bir zamanda veri tabanındaki veri, veri tabanı durumu olarak tanımlanır. Belirli bir veri tabanı durumunda, her şema veri tabanının o andaki örneklerine sahiptir. Herhangi bir veriyi değiştirdiğimizde veya eklediğimizde veri tabanı bir durumdan diğerine geçiş yapar.

Veri tabanı şeması ile veri tabanı durumu arasındaki fark çok önemlidir. Yeni bir veri tabanı tanımlandığında sadece onun şemasını VTYS'ne tanımlarız. O andaki veri tabanı durumu boş durumdur yani verisizdir. Veriyi ilk girdimiz andaki durum ise başlangıç durumudur.

5.2.1.4 VTYS Mimarisi

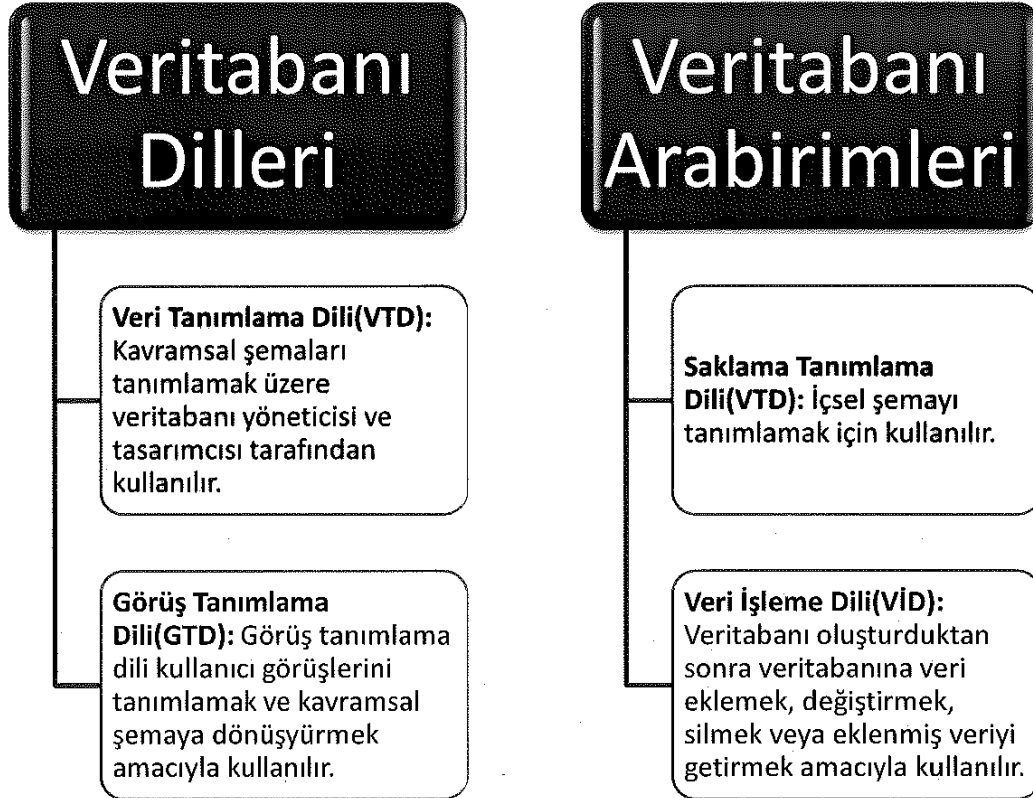
Bu mimarinin amacı kullanıcı uygulamaları ile fiziksel veri tabanını birbirinden ayırmaktır. Bu mimaride şemalar üç seviyede tanımlanabilir. Bu nedenle üç şema mimarisi olarak da anılır.

Pek çok VTYS bu üç düzeyi tam olarak birbirinden ayırmaz. Burada bu üç şemada da verinin tanımlayıcı olduğuna dikkat etmek gerekir. Sadece fiziksel düzeyde gerçek veri bulunur. Üç şema mimarisine dayalı VTYS'lerde her kullanıcı kendi dış şemasıyla ilgilidir. Bundan dolayı VTYS, dış şemasında belirlenmiş bir isteği kavramsal şemaya dönüştürmek ve daha sonra da iç şemaya dönüştürmek gereklidir. İstek ve sonuçları, düzeyler arasındaki dönüştürme işlemine **eşleme** denir.



5.2.1.5 Veri Tabanı Dilleri Ve Arabirimleri

Veri tabanı tasarımı tamamlandıktan sonra bir VTYS seçilir. Günümüzde bu diller birbirlerinden bağımsız olarak kullanılmamaktadır. Bunun yerine daha geniş, ayrıntılı ve tüm bu fonksiyonları yerine getiren tümleşik bir dil kullanılmaktadır. Bunun tipik bir örneği SQL verilebilir.



İki tip VİD vardır;

- **Yüksek Düzeyli VİD :** Bir bilgisayar terminalinden etkileşimli olarak kullanılabildiği gibi bir programlama dili içerisine de yerleştirilebilir.
- **Düşük Düzeyli VİD :** VİD, bir programlama dili içerisine gömülü olarak çalışır.

VTYS arabirimleri şunlardır;

- **Menü-Tabanlı:** Kullanıcıya çeşitli seçenekler sunulur.
- **Grafiksel Tabanlı:** Kullanıcıya veri tabanı şeması diyagramı halinde sunulur. Kullanıcı bu diyagram yardımıyla sorgu belirtebilir.
- **Form Tabanlı:** Kullanıcıya doldurmak üzere bir form sunulur.
- **Doğal Dil:** İngilizce yazılan sorguları kabul eder ve anlamaya çalışırlar.

5.2.1.6 Veri Tabanı Sistemi Ortamı

VTYS, karmaşık bir yazılım sistemidir. Bunda bir VTYS'yi oluşturan yazılım bileşenlerden söz edilmektedir. Tipik bir VTYS yazılımı bileşenleri şu şekildedir;

- VTYS Kataloğu
- Veri tabanı derleyicisi (VTD)
- Veri yöneticisi
- VİD derleyicisi
- Veri tabanı işlemcisi
- Yardımcı yazılımlar

Veri tabanı ve VTYS kataloğu genellikle disk üzerinde saklanır. Diske erişim öncelikle işletim sistemi tarafından kontrol edilse de Yüksek-Düzeyde saklanmış veri yöneticisi modülü disk üzerinde bulunan VTYS bilgilerine erişimi denetler. Veri tabanı derleyicisi, şema tanımlarını işler ve şema belirtimlerini(meta-veri) VTYS kataloğunda saklar. Katalog;

- Dosya adı,
- Veri adedi,
- Her dosya için saklama detayları,
- Şemalar arası mapping bilgileri
- Sınırlamalar

gibi bilgiler içerir. VTYS yazılım modülleri bu bilgilere gereksinim duyduğunda kataloğa erişmek durumundadır.

İşletim ortamındaki veri tabanı işlemcisi, işletim sırasında veri tabanına erişimi yönetir. Erişim veya güncelleme işlemlerini alır ve veri tabanı üzerinde gerçekleştirir.

Diske erişim, veri yöneticisi tarafından gerçekleştirilir. Sorgu derleyicisi etkileşimli olarak girilmiş yüksek düzeyli sorguları gerçekleştirir. Sorgu derlemesi şu şekilde gerçekleştirilir; her sorgu öncelikle çözülür, analiz edilir ve işletim zamanı işlemcisinin bu isteği gerçekleştirmesi için çağrı yapılır.

Ön derleme VİD komutlarını bir programlama dilinde yazılmış uygulama programından alır. Daha sonra bu komutlar VİD derleyicisine kaynak kodun oluşturulması amacıyla gönderilir.

Yükleme, yedekleme, performans ölçme, sıralama, veri sıkıştırma, vb. fonksiyonları yerine getirmek amacıyla veri tabanı yardımcı yazılımları bulunur.

5.2.1.7 VTYS'nin Sınıflandırılması

Sınıflandırmada kullanılan veri modelleri şunlardır;

- **İlişkisel Model:** veri tabanı tablo yığınının oluşmuştur. Her bir tablo bir dosya olarak saklanabilir.
- **Ağ Modeli:** veri kayıt ve küme tipleri olarak gösterir.
- **Hiyerarşik Model:** veri ağaç yapısında gösterilir.
- **Nesne Yönetimli Model:** veri tabanı nesneleri, özellikleri ve işlemleri biçiminde gösterilir.

5.2.1.8 Hazır Program Kütüphane Dosyaları

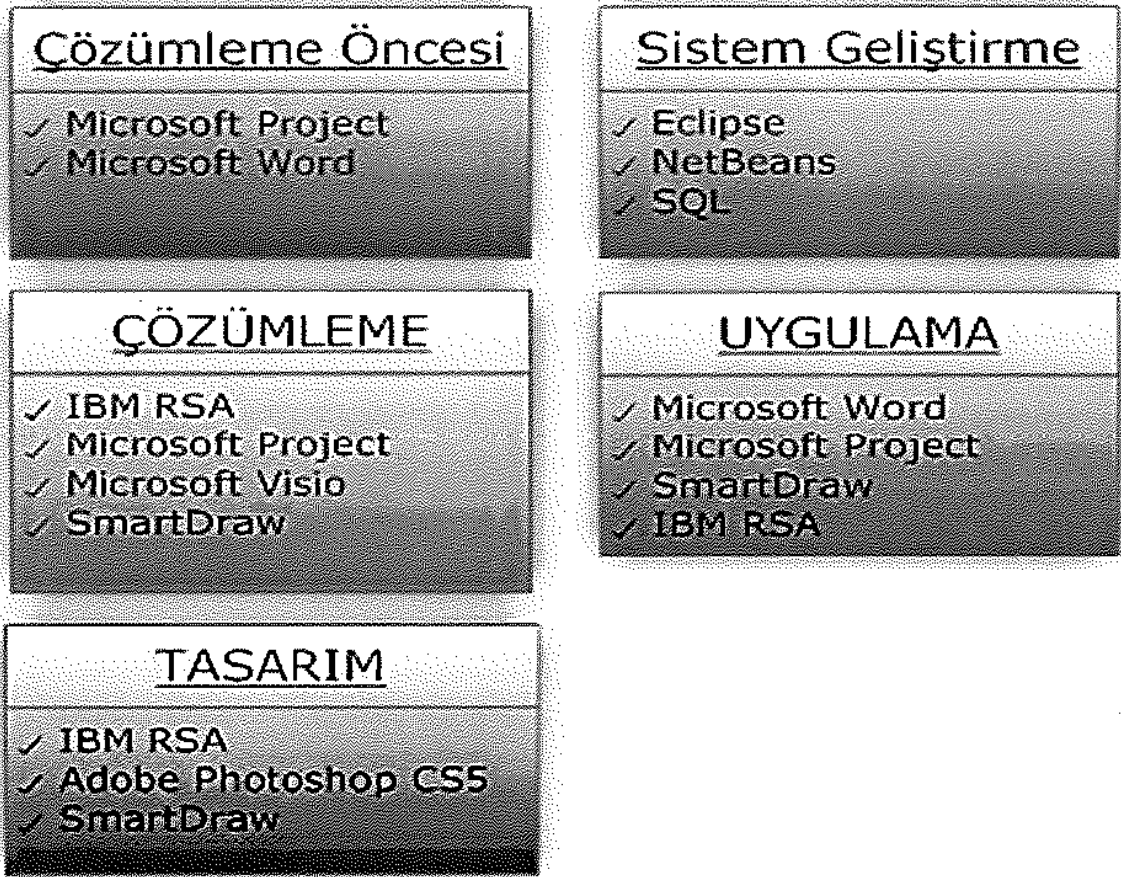
Hemen hemen tüm programlama platformlarının kendilerine özgü hazır program kütüphaneleri bulunmaktadır. Söz konusu kütüphaneler;

- Hemen hemen tüm uygulamalarda kullanılabilecek ortak program parçaları (OCX, VBX, vb.)
- Sistem yönetimine ilişkin yazılımları içerir

Söz konusu yazılımlar program geliştirme hızını oldukça arttırmaktadır. Günümüzde bu tür kütüphanelerin edinilmesi, internet sayesinde oldukça kolaylaşmıştır.

5.2.1.9 Case Araç Ve Ortamları

Günümüzde bilgisayar destekli yazılım geliştirme ortamları(CASE) oldukça gelişmiş durumdadır. CASE araçları, yazılım üretiminin hemen her aşamasında (planlama – çözümleme – tasarım – gerçekleştirim – sınama) üretilen bilgi ya da belgelerin bilgisayar ortamında saklanması, bu yolla kolay erişilebilir ve yönetilebilir olmasını olanaklı kılar. Hemen hemen tüm CASE araçları, belirli bir standarda uygun olarak geliştirme yapar. Bu yolla yapılan üretimin yüksek kalitede olması sağlanır.



5.3 Kodlama Stili

Hangi platformda geliştirilirse geliştirilsin, yazılımın belirli bir düzende kodlaması yazılım yaşam döngüsünün uygulama boyutu açısından oldukça önem taşımaktadır. Yazılım ya da bilgi sistemleri, doğaları gereği durağan değildir. Yazılımın gerektirdiği güncel değişikliklerin ilgili yazılıma da aktarılması gerekir. Bu nedenle yazılımın kodlarına zaman zaman başvurmak, yeni kod parçaları eklemek ya da var olan kodlarda değişiklikler yapmak yazılım yaşam döngüsünün işletimsel boyutunun en önemli işlevlerinden biridir. “Bakım Programcısı” kavramı bu tür gereksinimlerden doğmuştur. Bakım programcısının temel görevleri; var olan yazılıma ilişkin kodlar dâhil üretilmiş tüm bilgi ve belgeleri incelemek ve yazılım üzerinde değişiklikler yapmak biçiminde özetlenebilir. Kolay okunabilir ve anlaşılabilir kodları olmayan yazılımın bakımı oldukça zorlaşır ve büyük maliyetlere ulaşır.

Yazılım Kod stili konusunda herhangi bir kabul edilmiş standart bulunmamaktadır. Bu konuda; yazılım geliştiren ekiplerin, kodlama aşamasına başlamadan kodların düzeni konusundaki standartlarını ya da kurallarını geliştirmeleri ve bu kuralların uygulamaya geçmesini sağlamaları önerilmektedir.

5.3.1 Açıklama Satırları

Bir program parçasını anlaşılabilir kılan en önemli unsurlardan biri, bu program kesiminde içerilen açıklama satırlarıdır. Her bir program modülü içerisine;

- Modülün başlangıç açıklamaları
- Modül kod açıklamaları
- Boşluk satırları

Eklenmelidir.

Her bir modülün temel işlevleri, yazarın kişi, vb. bilgiler ilgili modülün en başına modül başlangıç açıklama satırları olarak eklenmelidir. Aşağıda C programlama dili için kullanılabilecek bir modül başlangıç açıklama satırında bulunması gerekenler şablon biçiminde verilmektedir.

```

                                Açıklama Satırı
/*****
* (c)
*****/
*Modül Adı:Sosyal Yardımlaşma Ve Dayanışma Otomasyon Sistemi
*Kütük Adı:SYDOS
*İşlevi:Yardımlaşma Ve Dayanışma
*Desteklenen Platformlar:Internet bağlantısının olduğu her yer
*Desteklenen Derleyiciler:NetBeans,Eclipse...
*Taşınabilirlik konuları:-
*Kullanılan İşlevler:-
*Hatalar:-
*Bilinen Düzeltilmemiş yanlışlar:-
*Programcı:-
*Tarihçe Günlüğü:
*İsim                                Tarih                                Açıklama
*****/
```

Bir programın karmaşıklığını arttıran en önemli bileşenler, program içerisinde kullanılan denetim yapılarıdır (koşullu deyimler, döngü deyimleri). Bu tür deyimlerin hemen öncesinde bu denetim işleminin açıklamasını ve bu deyimde olabilecek olağan dışı durumları içeren kod açıklama satırları koyulmalıdır.

Program kodu içerisinde; kodların görünebilirliğini ve anlaşılabilirliğini arttırmak amacıyla yeterli oranda boşluk satırı koyulmalıdır. Boşluk satırları programın etkinliğine (bellek kullanımı, performans, vb.) hiçbir olumsuz etki yapmaz. Bu nedenle gerek kod satırları arasına uygun olarak, gerekse bir satırın kendi içerisinde yeterli boşluk bırakılması önerilmektedir.

5.3.2 Kod Biçimlemesi

Kod biçimlenmesi, açıklama satırlarına olan ihtiyacı azaltır. Kod biçimlemesinde önemli olan az satır değil kodun okunabilirliğidir.

Sistemimizde programın okunabilirliğini artırmak ve anlaşılabilirliğini kolaylaştırmak amacıyla açıklama satırlarının kullanımının yanı sıra belirli bir kod yazım düzeni de kullanılmıştır.

```
VeriTabani vt=new VeriTabani();

Islemler islem=new Islemler();

this.kulBilgisi=islem.getKullaniciBilgisi();

int id=kulBilgisi.getKulId();

try{

    vt.baglan();

    this._anketListesi=vt.kisiyeAitAnketGetir(id);

    vt.baglantiyiKes();

}catch(Exception ex){

    return null;

}

return "anketlerim";
```

Yukarıda da görüldüğü gibi kodlama stili her bir alt dallarda başlama ve bitiş hizalamasına dikkat edilerek kodlama yapılmalıdır.

5.3.3 Anlamlı İsimlendirme

Anlamlı isimlendirme de kullanılan önemli teknik, hangi değişkenlere hangi modüllerle ilgili olduklarının belirtilerek adlandırılmasıdır. Kodların okunabilirliğini ve anlaşılabilirliğini sağlayan önemli unsurlardan biri de kullanılan ve kullanıcı tarafından belirlenen belirteçlerin (değişken adları, kütük adları, veri tabanı tablo adları, işlev adları, yordam adları, vb.) anlamlı olarak isimlendirilmesidir.

İsimlendirme yöntemi uygulamayı geliştirenler tarafından çözümleme-tasarım aşamalarında belirlenmeli ve gerçekleştirim aşamasında uygulanmalıdır. Kod incelenirken en azından ilk bakışta bilgilerin yalnızca değişken adına bakılarak anlaşılabilmelidir. Anlamlı isimlendirmede belirtilmesi gerekenler;

- Hangi değişkenin hangi verileri kapsadığı hangi tablolardan kullanıldığı
- Hangi değişkenin dışarıdan girildiği
- Hangi değişkenin çıktı olarak gösterileceği
- Hangi değişkenler sadece ara değişkenler olarak kullanılacağı
- Hangi değişkenlerin yazıcı çıktılarında görülmek üzere kullanıldıkları
- Hangi değişkenlerin yalnızca ilgili kod içerisinde ara değişkenler olarak kullanıldıkları
- Hangi değişkenlerin klavye ve ekran aracılığı ile değer aldıkları

5.3.4 Yapısal Programlama Yapıları (YPY)

Yapısal Programlama Yapıları, okunabilirlik ve anlaşılabilirlik bakımından önemlidir. Üç dala ayrılan YPY, yazılımın kodlanması sırasında sık sık kullanılacaktır. Program kodlarının; okunabilirlik, anlaşılabilirlik, bakım kolaylığı gibi kalite etmenlerinin sağlanması ve program karmaşıklığının azaltılması amacıyla YPY kullanılarak yazılması önemlidir. Yapısal Programlama Yapıları temelde; içinde “84 oto” deyimini bulunmayan, “tek giriş ve tek çıkış” öbeklerden oluşan yapılardır. Teorik olarak herhangi bir bilgisayar programının, yalnızca Yapısal Programlama Yapıları kullanılarak yazılabileceği kanıtlanmıştır.

Bu yapıların her biri “tek giriş ve tek çıkış” yapılardır. Programlar, yalnızca bu yapılar kullanılarak kodlandığında ve uygun açıklama satırları ile desteklendiğinde; program bakımı kolaylaşmakta ve geliştirme hızlanmaktadır.

Üç temel Yapısal Programlama Yapısı şunlardır;

- **Ardışık İşlem Yapıları:** Herhangi bir koşula bağlı olmaksızın birbiri ardına uygulanması gereken işlemler olarak tanımlanır. Hemen her türlü programlama dilinde bulunan; aritmetik işlem deyimleri, okuma/yazma deyimleri bu tür yapılara örnek olarak verilebilir.
- **Koşullu İşlem Yapıları:** 70’li yılların ortalarından sonra gelen programlama dillerinin hemen hepsinde, bu yapılar doğrudan desteklenmektedir. Üç tür Koşullu İşlem Yapısı bulunmaktadır;
 - Tek koşullu işlem yapısı → if-then
 - İki koşullu işlem yapısı → if-then-else
 - Çok koşullu işlem yapısı → case-when
- **Döngü Yapıları:** Belirli bir koşula bağlı olarak ya da belirli sayıda, bir veya daha çok yinelenen işlemler için kullanılan yapılardır. Temelde üç tür Döngü Yapısı bulunmaktadır;
 - Belirli sayıda yinelenen işlemler için kullanılan yapılar → for

- Bir koşula bağlı olarak, sıfır ya da birden çok kez yinelenen işlemler için kullanılan yapılar → while-end
- Bir koşula bağlı olarak, bir ya da birden çok kez yinelenen işlemler için kullanılan yapılar → repeat-until

5.4 Program Karmaşıklığı

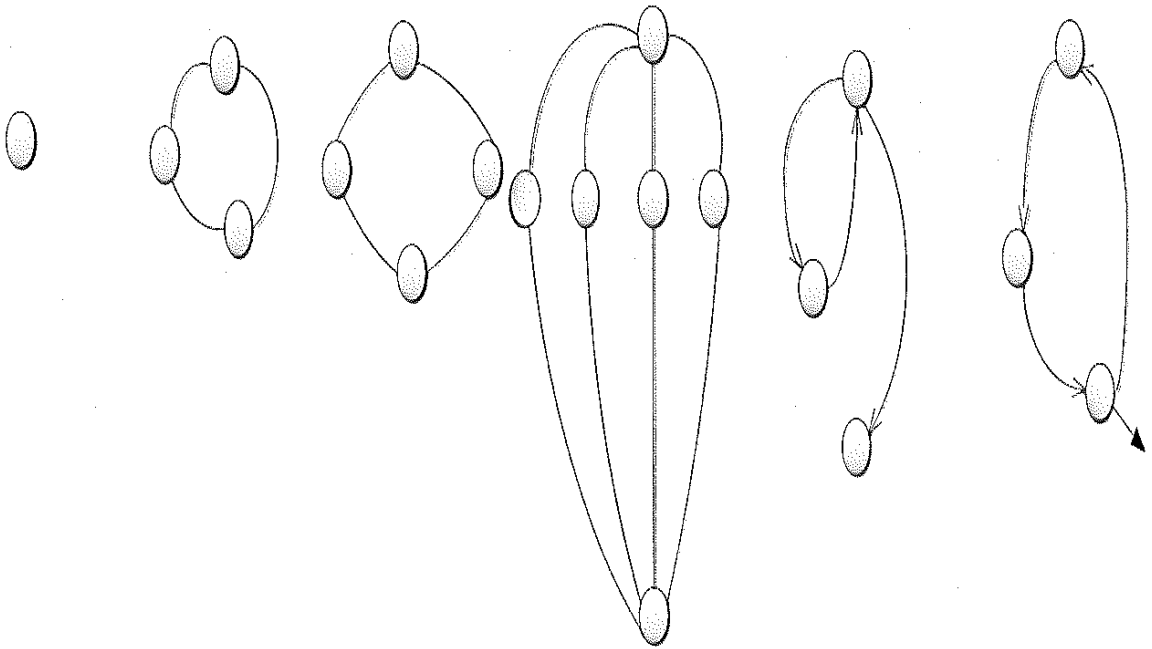
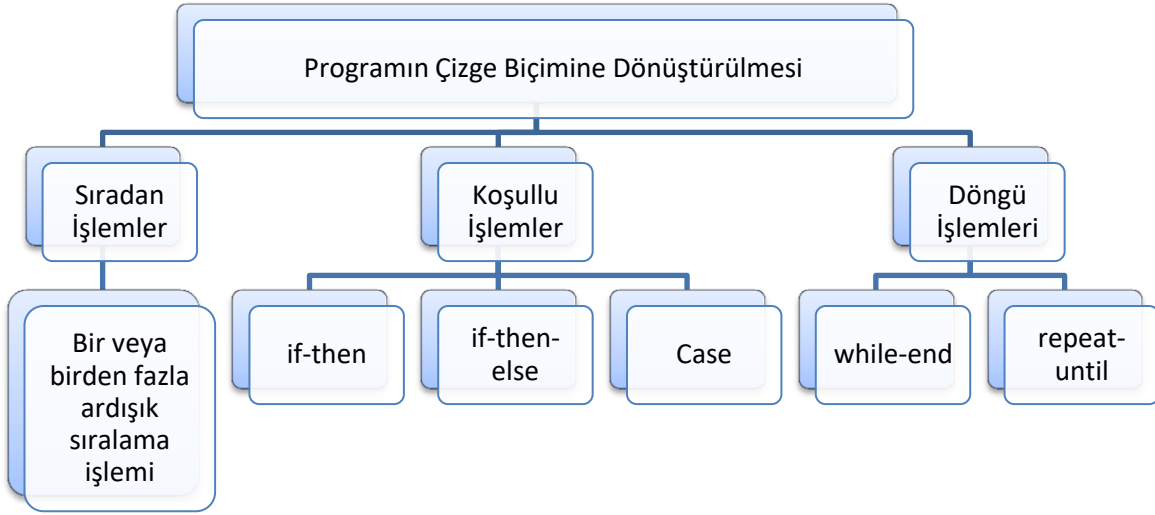
Program karmaşıklığını ölçmek için bir çok teorik model geliştirilmiştir. Bu modellerin en eskisi ve yol göstericisi McCabe karmaşıklık ölçütüdür. Söz konusu ölçüt 1976 yılında McCabe tarafından geliştirilmiştir. Bu konuda geliştirilen diğer ölçülerin çoğu, bu ölçütten esinlenmiştir.

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

1. Programın çizge biçimine dönüştürülmesi
2. McCabe karmaşıklık ölçütü hesaplanması



5.4.1 Programın Çizge Biçimine Dönüştürülmesi



5.4.2 McCabe Karmaşıklık Ölçütü Hesaplama

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

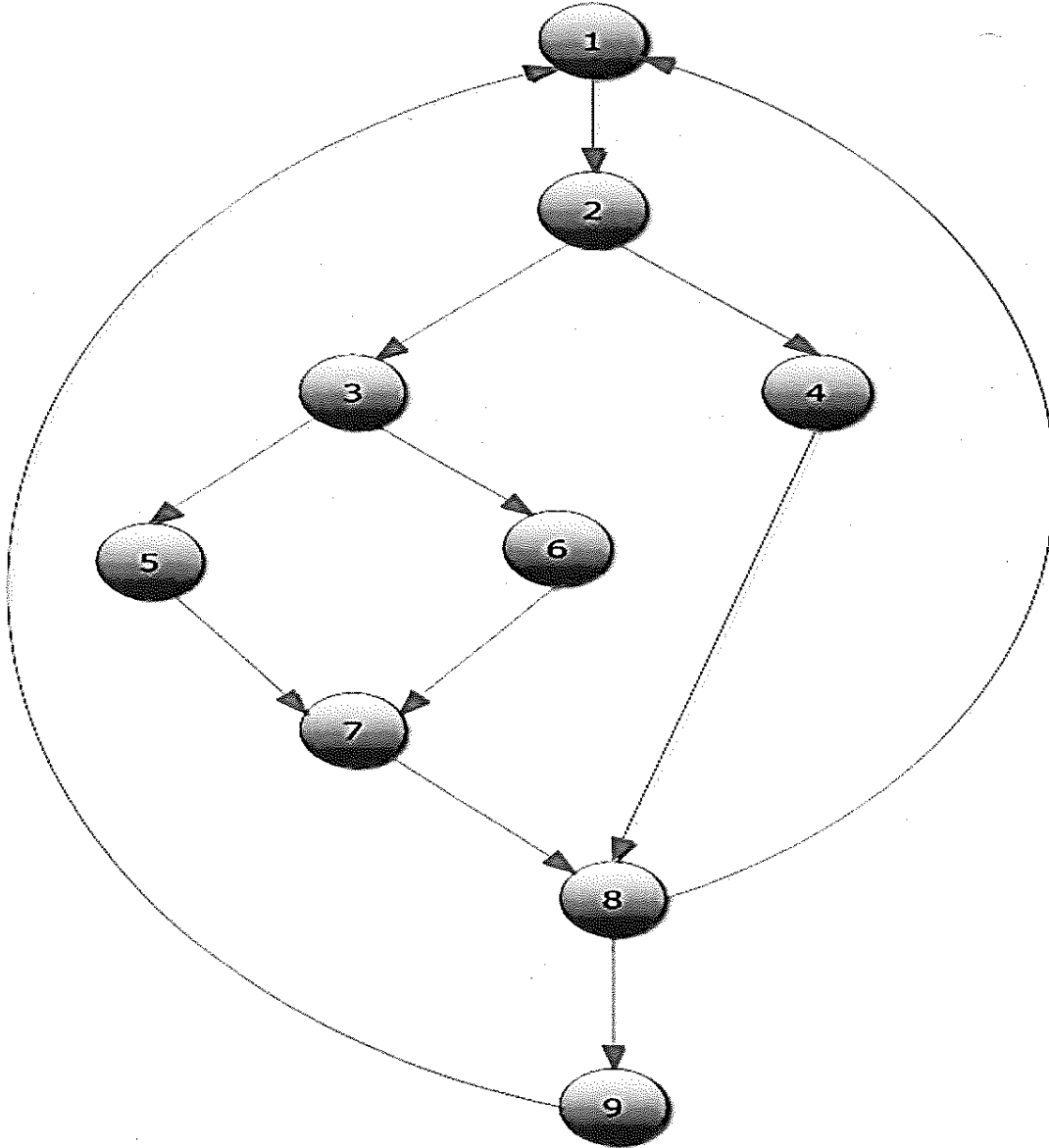
k=Diyagramdaki kenar çizgi sayısı

d=diyagramdaki düğüm sayısı

p= programdaki bileşen sayısı (ana program ve ilgili alt program sayısını göstermektedir, alt program kullanılmadı ise p=1, n tane alt program kullanıldı ise p=(n+1) olur)

Örnek

Olarak;



k=11 kenar sayısı

d=9 düğüm sayısı

p=1 bileşen sayısı

$V(G)=11-9+2=4$

5.5 Olağan Dışı Durum Çözümleme

Olağan dışı durumlar gerek kod yazım sürecinde gerekse testler sırasında gerçekleşebilir. Verilen hata kodlarının kısa sürede gözden geçirilmesi gerekir. Modüller olarak kodlanan programda bir modülde gerçekleşen hata diğerini etkilese bile çözümü kolay olacaktır. Sistemin çalışmasının; geçersiz, yanlış veri oluşumu veya başka nedenlerle istenmeyen bir biçimde sonlanmasına karşı önlemler alınmıştır. Hata yakalamak için sitemimizde try-catch kod blokları kullanılacaktır.

5.5.1 Olağandışı Durum Tanımları

Olağandışı durumlar, programlama dili tarafından tanımlı durumlar olabileceği gibi kullanıcı tarafından da tanımlanabilmektedir

5.5.2 Farklı Olağan Dışı Durum Çözümleme Yaklaşımları



- **Anında Durdurma:** Hata bulunduğunda program açıklayıcı bir ileti vererek sona erer. Bu iletinin yanı sıra 16'lık sistemde bir döküm de verir. MS Office paketlerinde alınan “This Program Performed an illegal Operation” iletisi ve ardından programın durması bu tür olağan dışı durum çözümleme yaklaşımına

örnek olarak verilebilir. Bu yaklaşımda, verilen hata iletisinin anlamlı olması ve programcıya, düzeltilebilmek amacıyla yol gösterici olması önemlidir.

- **Hata Kodu Verme:** İlgili program modülünde başarısız bir şekilde çıktığında, programın bir hata kodu vermesi biçiminde bir yaklaşımdır. Söz konusu kod “başarılı” veya “başarısız” durumu belirten mantıksal bir kod olabileceği gibi bazı uygulamalarda sayısal ya da metin biçiminde olabilir. Bu yaklaşımda ilgili programda da hata olduğu anlaşılabilen ancak hatanın hangi deyimde olduğu anlaşılamamaktadır.
- **Tanımlı Olağandışı Yordam Çalıştırma:** Programlama dili platformunun tanımladığı olağan dışı durum çözümleme olanakları kullanmak biçimindedir. COBOL programlama dilindeki “ON READ ERROR” deyimini gibi.
- **Hata Yordamı Çalıştırma (Olmayacak Değer Döndürme) :** Yordamın beklenmeyen bir değer geri döndürmesi gibi bir hesaplama değildir. Örneğin yaş hesaplama yapıp, yaş bilgisini geri döndüren bir programın eksi değer döndürmesi gibi.

5.6 Kod Gözden Geçirme

Hiç kimse, önceki sürümlerini gözden geçirmeden ve incelemeyen okunabilir bir program yazamaz. Hiçbir yazı, editörün onayını almadan basılmayacağı gibi hiçbir program da incelenmeden, gözden geçirilmeden işleme alınmamalıdır. Kod gözden geçirme ile program sınaama işlemlerini birbirinden ayırmak gerekir.

Program sınaama, programın işletimi sırasında ortaya çıkabilecek yanlış ya da hataları yakalamak amacıyla yapılır. Kod gözden geçirme işlemi 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 gireceğini 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 inceleme 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 çalışmayacağının belirlenmesidir. Gözden geçirme çalışmasının olası çıktıları;

P

***Programı olduğu gibi kabul etmek.**

R

O

G

R

A

M

***Programı bazı değişikliklerle kabul etmek.**

***Programı, önerilen değişikliklerin yapılmasından sonra tekrar gözden geçirmek için geri çevirmek.**

5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular

5.6.2.1 Öbek Ara Yüzü

- Her öbek tek bir işlevsel amacı yerine getiriyor mu?
- Öbek adı, işlecini açıklayacak biçimde anlamlı olarak verilmiş mi?
- Öbek tek giriş ve tek çıkışlı mı?
- Öbek eğer bir işlev ise parametrelerinin değerini değiştiriyor mu?

5.6.2.2 Giriş Açıklamaları

- Öbek, doğru biçimde giriş açıklama satırları içeriyor mu?
- Giriş açıklama satırları, öbeğin amacını açıklıyor 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 tamamlıyor mu?
- Giriş açıklama satırları, öbeği yazan kişi ve yazıldığı tarih ile ilgili bilgileri içeriyor mu?
- Her paragrafı açıklayan kısa açıklamalar var mı?

5.6.2.3 Veri Kullanımı

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

5.6.2.4 Öbeğin Düzenlenişi

- Algoritmalar istenen işlevleri karşılıyor mu?
- Ara yüzler genel tasarımla uyumlu mu?
- Mantıksal karmaşıklık anlamlı mı?
- Veri yapısı çözümleme çalışması sırasında elde edilen veri modeli ile uyumlu mu?
- Belirlenen tasarım standartlarına uyulmuş mu?
- Hata çözümleme tanımlanmış mı?
- Tasarım, kullanılacak programlama diline uygun mu?
- İşletim sistemi ve programlama diline yönelik kısıtlar ya da özellikler kullanılmış mı?
- Bakım dikkate alınmış mı?

5.6.2.5 Sunuş

- Her satır, en fazla bir deyim içeriyor mu?
- Bir deyim birden fazla satıra taşması durumunda bölünme anlaşıla bilirliliği kolaylaştıracak biçimde anlamlı 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

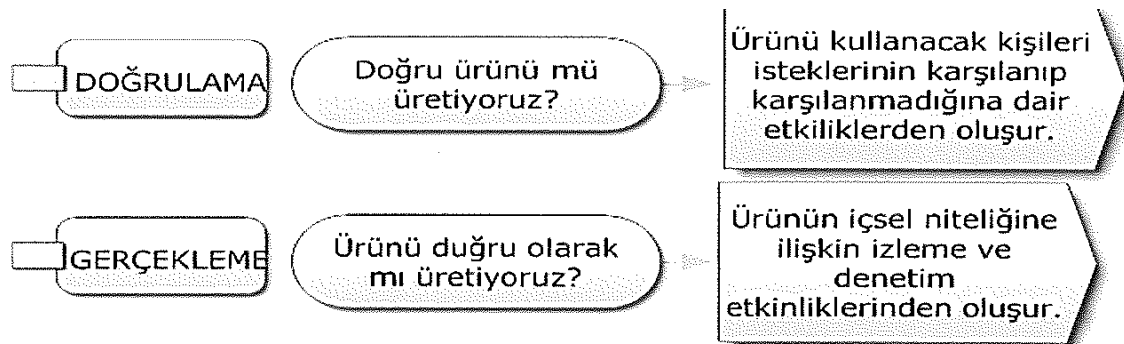
Yazılım geliştirme karmaşık bir süreç olduğundan, hataların ortaya çıkması kaçınılmazdır. Hataların erken belirlenmesine katkıda bulunacak ve yazılım kalitesini arttıracak metotlar;

- Yazılım, yaşam döngüsünün her aşamasında hatalara karşı sınanması
- Gereksinimler arasındaki tutarsızlıkların giderilmesi
- Çözümleme şeması ile uygulama alanı arasındaki uyumsuzlukların giderilmesi
- Tasarım hatalarının düzeltilmesi
- Çalışma anı hatalarının giderilmesi
- Yaşam döngüsünde ilerledikçe, hataların düzeltilmesi
- İyi bir sına yaklaşımı

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

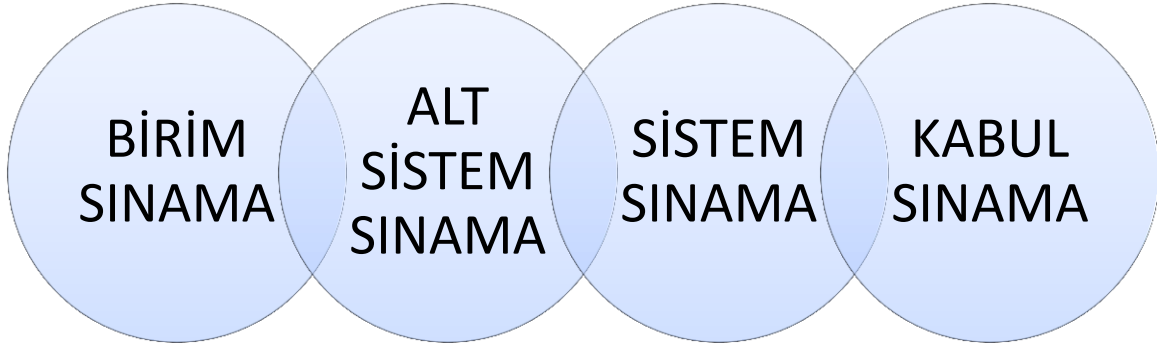
- Yazılım belirtilerinin ve proje yaşam sürecindeki her bir etkinlik alınan çıktıların; tamamı doğru, açık ve önceki belirtileri tutarlı olarak betimler durumda olduğunun doğrulanması
- Proje süresince her bir etkinlik ürününün teknik yeterliliğinin değerlendirilmesi ve uygun çözüm elde edilene kadar aktivitenin tekrarına sebep olması
- Projenin bir aşaması süresince geliştirilen anahtar belirtilerin önceki belirtilerle karşılaştırılması
- Yazılım ürünlerinin tüm uygulanabilir gerekleri sağlandığının gerçekleştirilmesi için sınamaların hazırlanıp yürütülmesi

Biçiminde özetlenebilir.



Doğrulama ve Gerçekleme Arasındaki Temel Fark

6.2 Sınama Kavramları

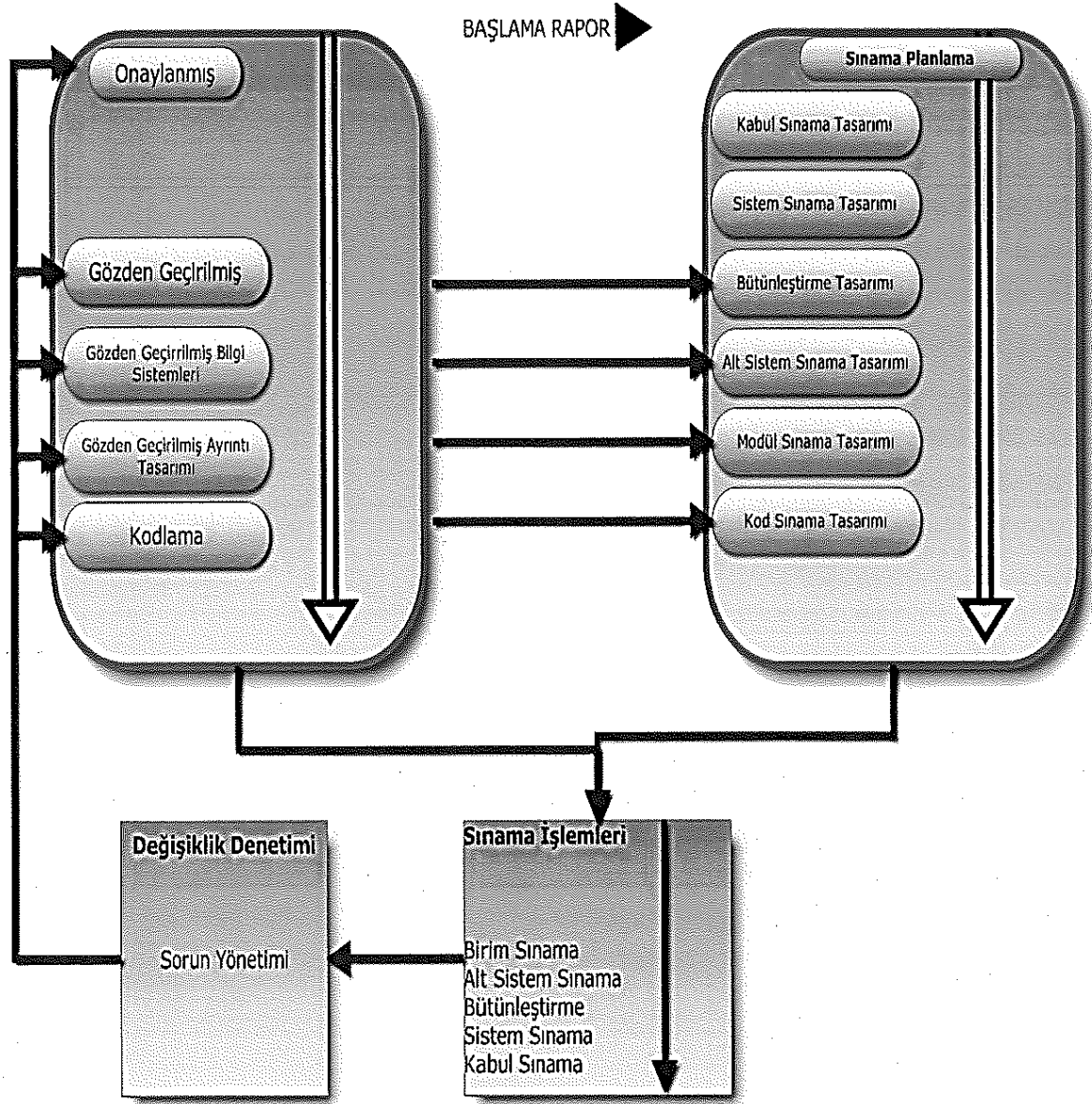


- **Birim Sınama:** Bağlı oldukları diğer sistem unsurlarından tümüyle soyutlanmış olarak birimlerin doğru çalışmalarının belirlenmesi amacıyla yapılır
- **Alt Sistem Sınama:** Alt sistemler, modüllerin bütünleştirilmeleri ile ortaya çıkarlar. Yine bağımsız olarak sınamaları yapılmalıdır. Bu aşamada en çok hata ara yüzlerde bulunmaktadır. Bu yüzden ara yüz hatalarına doğru yoğunlaşılmalıdır.
- **Sistem Sınama :** Üst düzeyde, bileşenlerin sistem ile olan etkileşiminde olan hatalar aranmaktadır. Ayrıca belirtilen ihtiyaçların doğru yorumlandıkları da sınanmalıdır.
- **Kabul Sınama:** Çalıştırılmadan önce sistemin son sınamasıdır. Artık yapay veriler yerine gerçek veriler kullanılır. Bu sınama türü alfa sınaması veya beta sınaması olarak da bilinir.

6.3 Doğrulama Ve Geçerleme Yaşam Döngüsü

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 tasarımlarını içerir. Gerçekleştirim aşamasının sonunda ise söz konusu planlar uygulanır.

Uygulama sonucu elde edilen bulgular, Yazılım Doğrulama ve Geçerleme raporları biçiminde sürekli olarak raporlanır. Bu bilgiler, değişiklik denetim sistemi ve sorun yönetim sistemlerinde girdi olarak kullanılır. Değişiklik Denetim Sistemi, sınama sonucu elde edilen bulgular ve bunlara karşı sorun yönetimi tarafından alınan önlemler izlenir.



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. Her mühendislik ürünü, iki yoldan biri ile sınanır. Sistemin türüne yönelik işlevlerin doğru yürütüldüğünün (kara kutu – black box) veya iç işlemlerin belirtilmelere uygun olarak yürütüldüğünün bileşenler tabanında sınanması (beyaz kutu – white box).

Kara kutu sınamasında sisteme, iç yapısı bilinmeksizin gelişigüzel girdiler verilerek sınama yapılır. Beyaz kutu sınaması için ayrı bir başlık ayrılmıştır.

6.4.1 Beyaz Kutu Sınaması

Beyaz kutu sınaması tasarlanırken, birimin süreç belirtiminden yararlanılır. Yapılabilecek denetimlerin arasında şunlar bulunur;

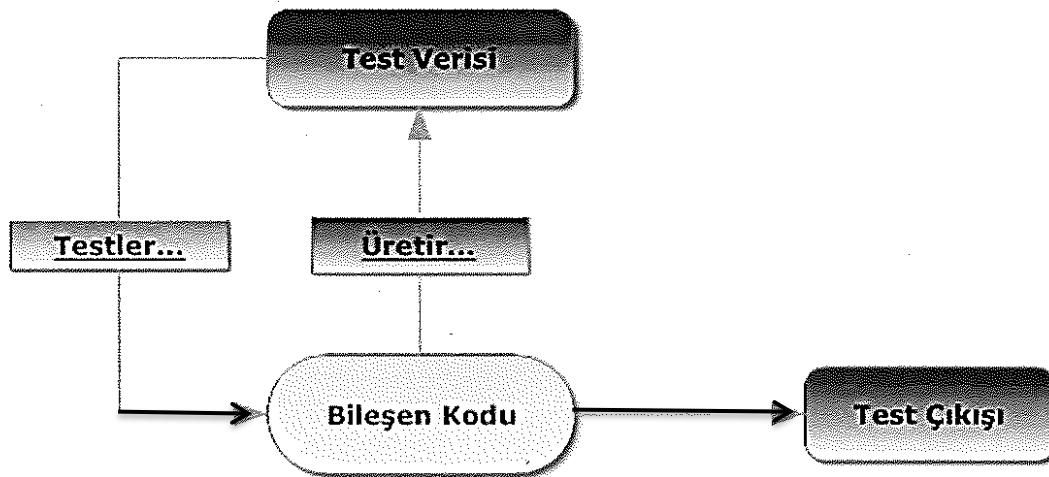
- Bütün bağımsız yolların en azından bir kere sınanması
- Bütün mantıksal karar noktalarında iki değişik karar için sınamaların yapılması
- Bütün döngülerin sınır değerlerinde sınanması
- İç veri yapılarının denenmesi

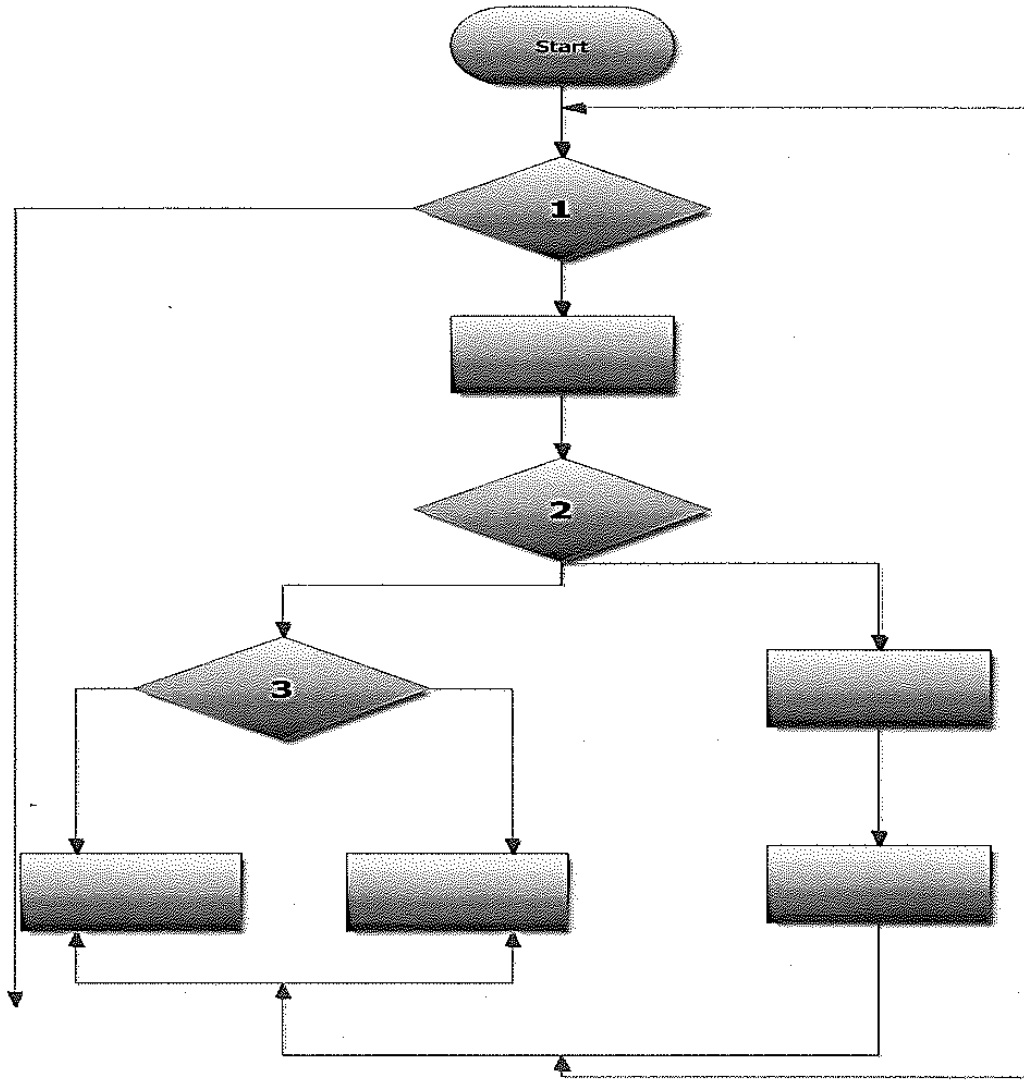
Sınamaları yürütürken sınırlı çabamızı yerinde kullanmamız gerekir. Bunun için hataların bazı özelliklerinin bilinmesinde yarar vardır;

- Bir program kesiminin uygulamada çalıştırılma olasılığı az ise o kesimde hata olması ve bu hatanın önemli olması olasılığı fazladır.
- Çoğu zaman, kullanılma olasılığı çok az olarak kestirilen program yolları aslında çok sıkça çalıştırılıyor olacaktır.
- Yazım hataları rastgele olarak dağılır. Bunlardan bazıları derleyiciler tarafından bulunur, bazıları da bulunmadan kalır.

6.4.2 Temel Yollar Sınaması

Daha önce çevrimsellik karmaşıklığı konusunda gördüğünüz hesap yöntemi ile bir programdaki bağımsız yollar bulunduktan sonra bulunan sayı kadar sınama yaparak programın her birimini bir şekilde sınamalara dahil etmiş oluruz. Bağımsız yolların saptanması için önce program çizgesel bir biçime çevrilir. Bunu yapmak için ise program iş akış şemaları diyagramları iyi bir başlangıç noktasıdır.





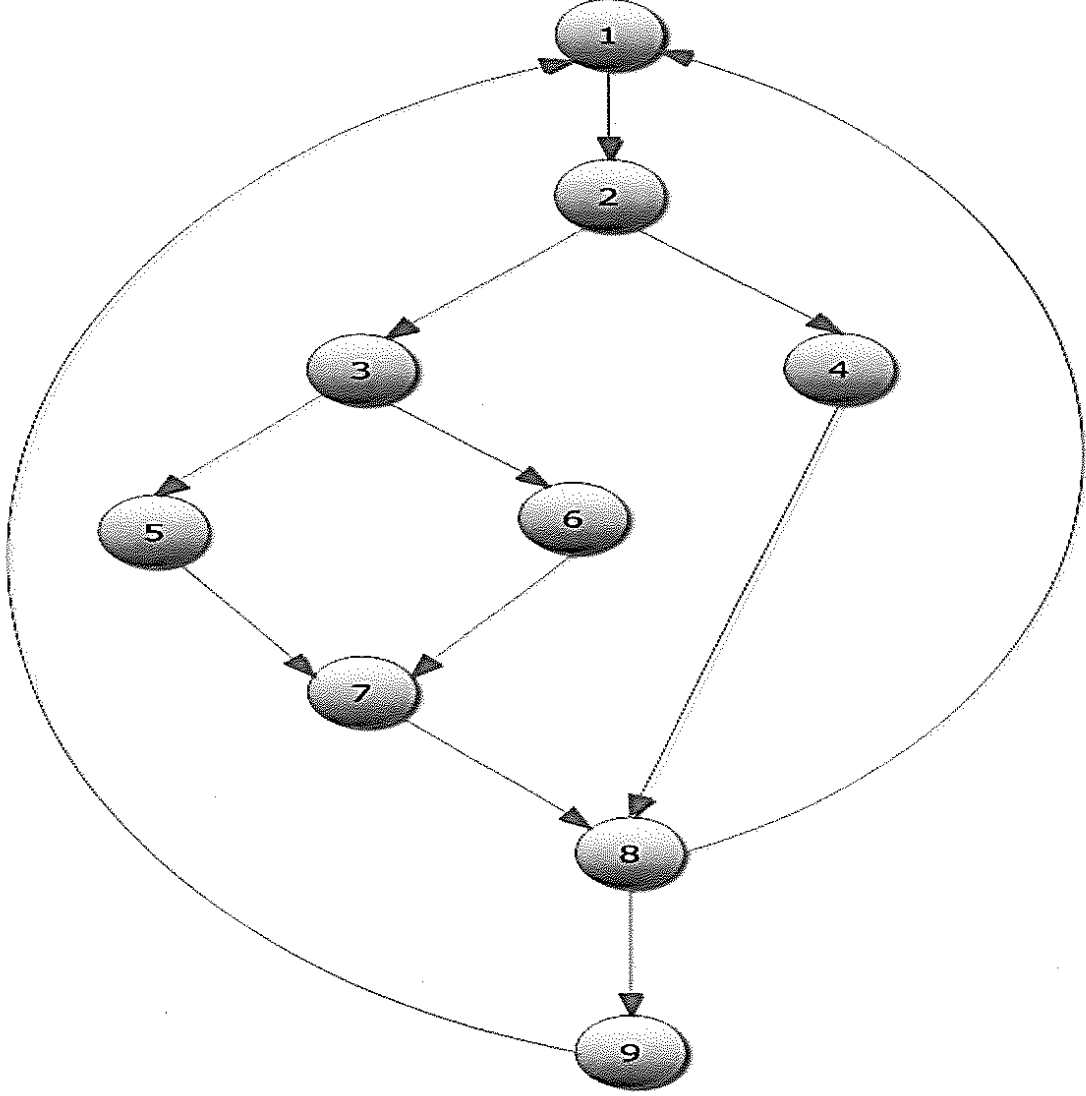
Program akış diyagramları(yukarıdaki şekil) matematiksel titizlik ile tanımlanmayan daha serbestçe program yapılarını alt düzey modelleyen çizimlerdir. Akış diyagramları ise Çizge Teorisi dalında kabul görecektir. Her çizgede olduğu gibi burada da düğümler ve dallar(veya kişiler) bulunur. Program akış diyagramından akış diyagramına geçmek için süreç kutuları ortadan kaldırılır, koşul baklavaları yerine düğümler çizilir ve her koşul düğümlerine karşılık olarak birleştirme düğümleri eklenir. Birleştirme düğümleri, koşul kollarının kapandığı noktaya konur. Artık bu çizge üzerinden temel yol sayısını da verecek olan çevrimsellik karmaşıklığı sayısını hesaplayabiliriz;

$$E - N + 2$$

$E \rightarrow$ Toplam dal sayısı

$N \rightarrow$ Toplam düğüm sayısı

Bağımsız temel yol sayısı kadar temel yollar, çizge üzerinde (sonunda programa yansıtılmak üzere) veya program üzerinde işaretlenir. Sonra bu yolların hepsinin koşturulacağı sınama programları tasarlanır.



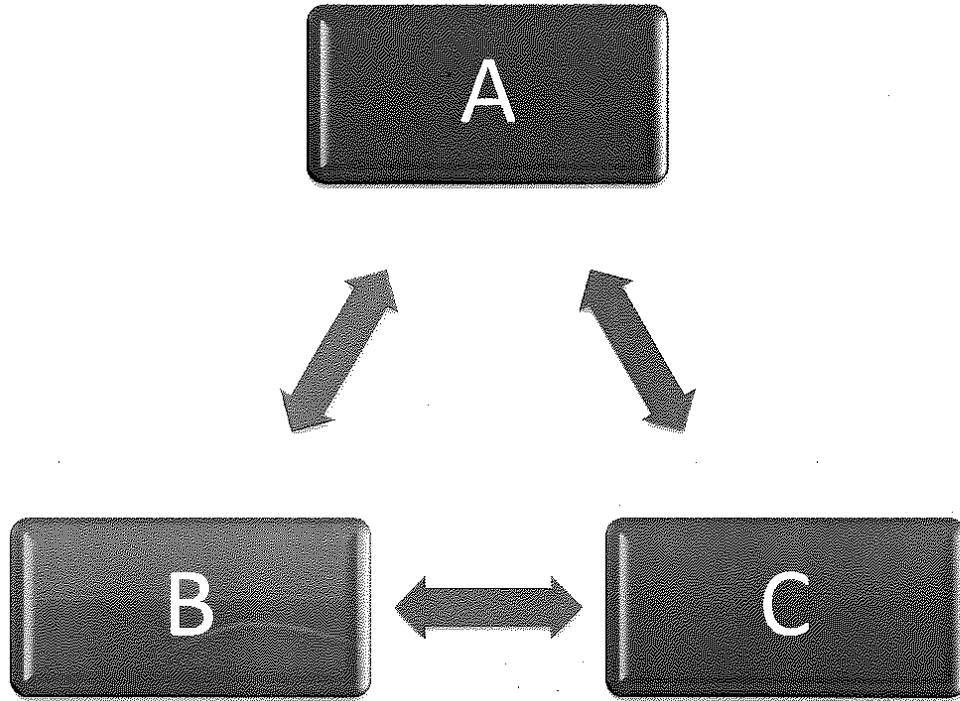
Hesaplama: $11-9+2=4$

Bunun anlamı da şöyle açıklanabilir; Çizgedeki her dal sınama işleminde kapsanmalıdır. Bu, sınama sırasında her işlemin çalıştırılması demektir. Her daim en az bir kere kapsanacağı ve en az sayıda yollar bulunmalıdır. Programa ortadan giremeyeceğimize göre de bu yollar başlangıç noktasından bitiş noktasına kadar uzanmalıdır. Son olarak da minimum sayıda yol ile bu şartları sağlamalıyız.

6.5 Sınama Ve Bütünleştirme Stratejileri

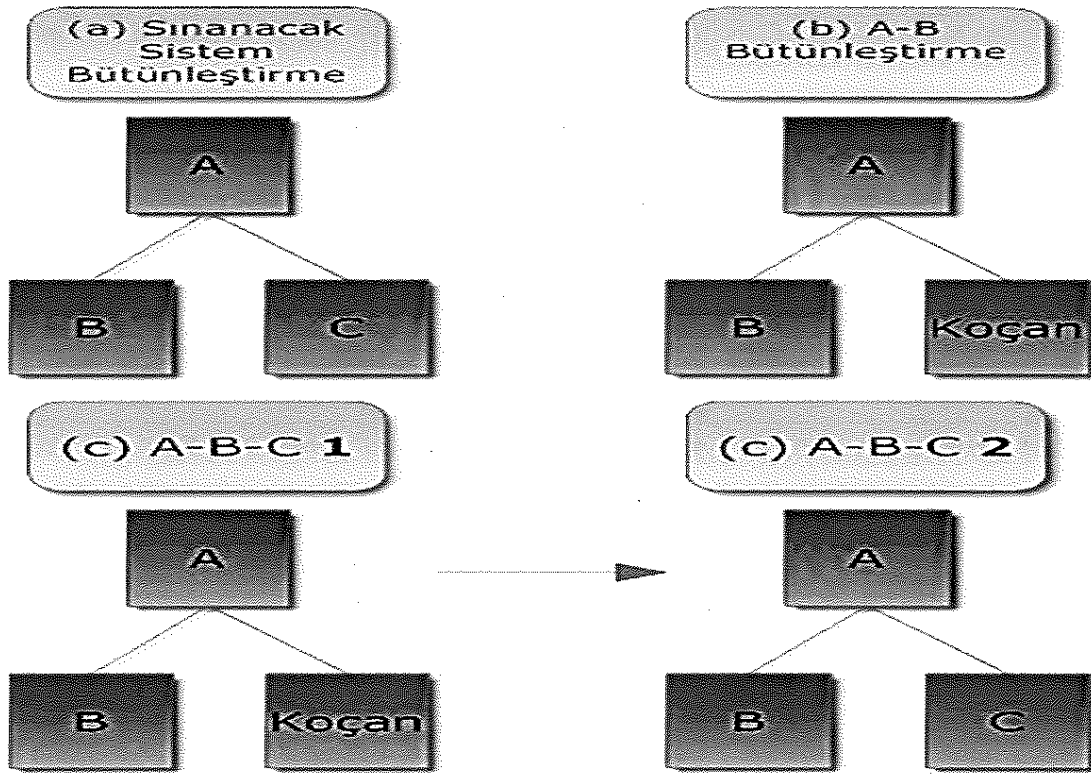
Genellikle sınama stratejisi, bütünleştirme stratejisi ile birlikte değerlendirilir. Ancak bazı sınama stratejileri bütünleştirme dışındaki sorunları hedefleyebilir. Örneğin, yukarıdan aşağı ve aşağıdan yukarı stratejileri bütünleştirme yöntemine bağımlıdır. Ancak işlem yolu ve gerilim sınamaları, sistemin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

Sınama ve Bütünleştirme Stratejilerinde Yukarıdan Aşağı sınama gerçekleştirilir. Yukarıdan aşağı bütünleştirmede önce sistemin en üst düzeyi daha sonra aşağı doğru olan düzeyler sınanır. Öncelikle A modülü incelenir eğer doğru sonuçlar veriyorsa B ve C modüllerine geçilir.



6.5.1 Yukarıdan Aşağı Sınama Ve Bütünleştirme

- Yukarıdan aşağıya bütünleştirmede önce sistemin üst düzeylerinin sınanması ve sonra aşağıya doğru olan düzeylere ilgili modüllerin takılarak sınanması söz konusudur.
- En üst noktadaki bileşen sılandıktan sonra alt düzeye geçilmelidir.
- Alt bileşenler henüz hazırlanmamışlardır. Bu sebepte koçanlar kullanılır. Koçan; bir alt bileşenin, üst bileşen ile ara yüzünü temin eden fakat işlevsel olarak hiçbir şey yapmayan çerçeve programlardır. İki temel yaklaşım vardır;
 - Dikey öncelikli bütünleştirme → en üst düzeyden başlanır ve aynı düzeydeki birimler bütünleştirilir
 - Derinlik öncelikli bütünleştirme → en üst düzeyden başlanır ve her dal soldan sağa olmak üzere ele alınır.



6.5.2 Aşağıdan Yukarıya Sınama Ve Bütünleştirme

Aşağıdan yukarı bütünleştirmede ise öncelikle yöntemin tersine uygulama yapılır. Önce en alt düzeydeki modüller sınanır ve bir üstteki birimle sınanması gerektiğinde bu üst bileşen, bir “sürücü” ile temsil edilir. Yine amaç, çalışmasa bile ara yüz oluşturacak ve alt bileşenin sınanmasını sağlayacak bir birim edinmektir. Bu kez kodlama, bütünleştirme ve sınama aşağı düzeylerden yukarı düzeylere doğru gelişir ve yukarı düzeylerde önce sürücü olarak yazılan birimler sonra gerçekleriyle yer değiştirerek o düzeyin birimleri/alt sistemi olurlar. Bütünleştirme yukarı doğru yapıldıkça daha az sürücü gereği duyulur.

Uygulamada, hem aşağıdan yukarıya hem de yukarıdan aşağı sınama stratejilerinin ikisinin birleştirildiği de olur. “Sandviç” adı verilen bu karma yaklaşımda hem üstten hem de alttan sınama etkinliği sürer.

- Sınamaya Başlanma Koşulları
- Girdilerin Hazır Olması
- Ortam Koşulları
- Kaynak Koşulları
- Sınama Tamamlama Kıstası
- Sınama Geçme-Kalma Kıstası
- Sınama Askıya Alınma Kıstası ve Sürdürme Gerekleri
- Sınama Sonuçları

Sınama planları şunlardır;

- Birim (modül) sınama planları
- Alt sistem sınama planları
- Bütünleştirme sınama planları
- Kabul sınama planları
- Sistem sınama planları

Her sınama planı; sınama etkinliklerinin sınırlarını, yaklaşımını, kaynaklarını ve zamanlamasını tanımlar. Plan neyin sınanacağını, neyin sınanmayacağını, sorumlu kişileri ve riskleri göstermektedir. Sınama planları, sınama belirtilmelerini içerir.

6.7 Sınama Belirtilmeleri

Sınama belirtilmeleri, bir sınama işleminin nasıl yapılacağına ilişkin ayrıntıları içerir.

Bu ayrıntılar 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 verilerinin elle hazırlanması çoğu zaman kolay olmayabilir ve zaman alıcı olabilir. bu durumda, otomatik sınama verisi üreten programlardan yararlanılabilir.

Sınama senaryoları, yeni sınama senaryosu üretebilmeye yardımcı olacak biçimde hazırlanmalıdır. Zira sınama belirtilmelerinin hazırlanmasındaki temel amaç, etkin sınama yapılması için bir rehber oluşturmaktır.

Sınama işlemi sonrasında bu belirtilmelere;

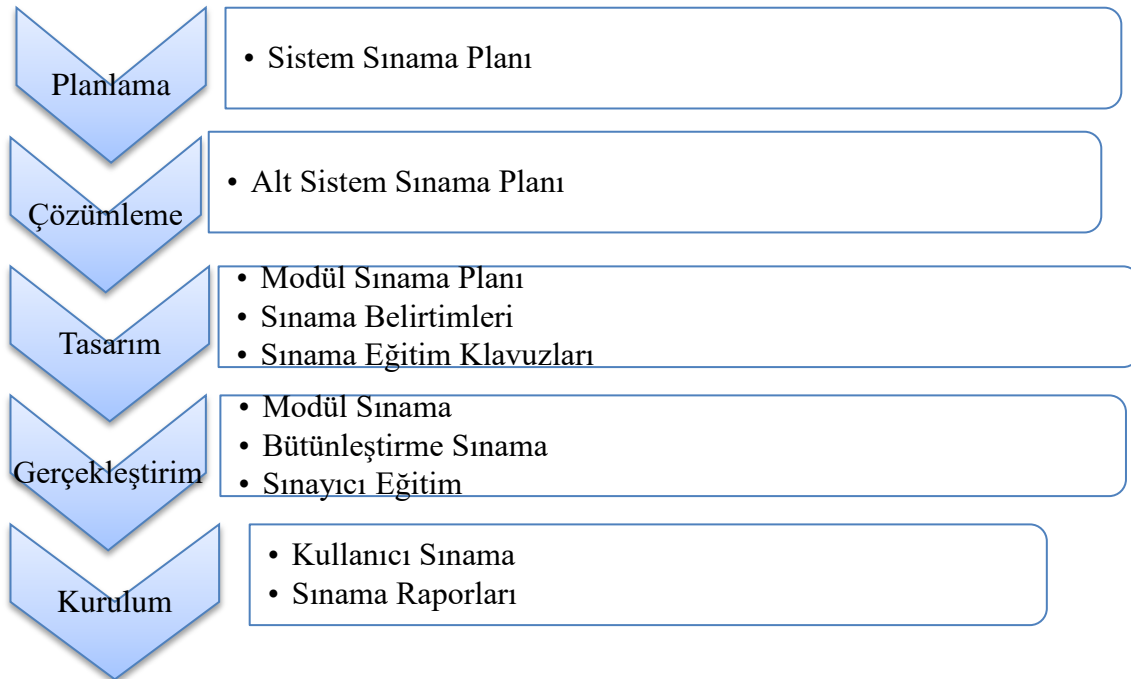
- Sınamayı yapan
- Sınama tarihi
- Bulunan hatalar ve açıklamalar

Türündeki bilgiler eklenerek sınaıa raporları oluşturulur.

Sınaıa raporları, sınaıa bitiminde imzalanır ve yüklenici ile iş sahibi arasında resmi belge niteliğı oluşturur.

6.8 Yaşam Döngüsü Boyunca Sınaıa Etkinlikleri

Sınaıa etkinlikleri; üretim aşamasının en başından başlayarak planlanır, ayrıntılanır ve raporlanır. Sınaıa etkinliklerinin yaşam döngüsü çekirdek adımlarındaki dağılımı gösterilmektedir. Yapılan işe ilişkin bilgi düzeyi arttıkça sınaıa planları soyut durumdan somut sınaıamalara dönüşür.



Hazırlanan sınaıa raporları, doğrulama ve geçerleme yaşam döngüsü işlemleri gereğı “sorun yönetimine iletilir. Bu bölümde hatalar kaydedilir ve bulunan hatalara karşı yapılacak işlemler planlanır. Sınamalar sırasında bulunan her bulgu ya da hata olarak belirtilen her durum gerçekte hata olmayabilir. Farklı sınaıacılarından biri, bir durumu hata olarak nitelerken diğeri aynı durumu doğru olarak değerlendirebilir. Bu nedenle sınaıa raporlarında hata olarak bildirilen her durum hemen düzeltilmek üzere ele alınmaz. Önce çözümlenir, kullanıcı çelişkileri giderilir ve gerçekten hata olduğuna karar verilirse düzeltilir. Söz konusu karar kullanıcı temsilcileri ile birlikte alınır.

Sınaıa sırasında bulunan her hata için Değışik Kontrol Sistemine (DKS), “Yazılım Değışiklik İsteğı” türünde bir kayıt girilir. Hatalar, DKS kayıtlarında aşağıdaki gibi gruplara ayrılabilir;

- **Olumsuz Hatalar:** Projenin gidişini bir ya da birden fazla aşama gerileten veya düzeltilmesi mümkün olmayan hatalardır.
- **Büyük Hatalar:** Projenin gidişini kritik olarak etkileyen ve önemli düzeltme gerektiren hatalardır.
- **Küçük Hatalar:** Projeyi engellemeyen ve giderilmesi az çaba gerektiren hatalardır.
- **Şekilsel Hatalar:** Heceleme hatası gibi önemsiz hatalardır.

Her sınama bitiminde, o sınama için tüm ilgili Değişiklik Kontrol Sistemi kayıtları kullanılarak ayrıntılı raporlar alınabileceği gibi genel hata istatistikleri de elde edilebilir.

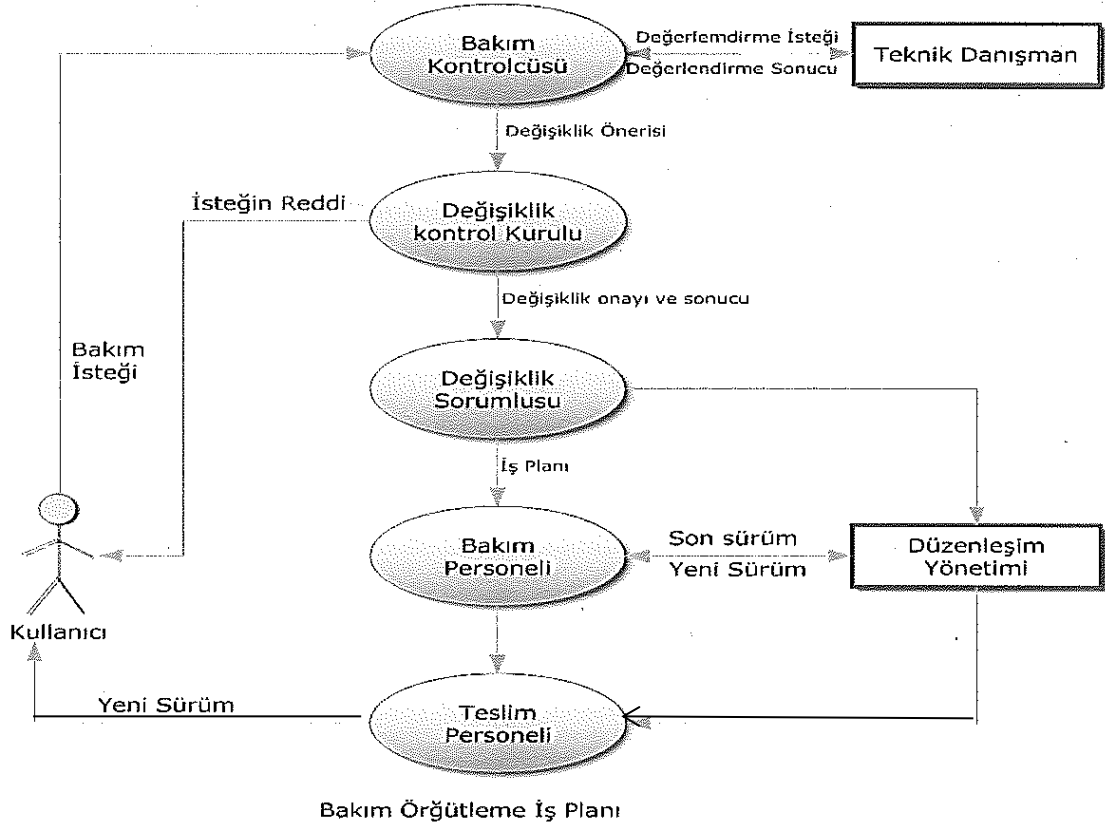
7. Bakım

7.1 Giriş

Bilgisayar tabanlı sistemlerin tasarlanıp geliştirilmesinde ve kullanıcıya tesliminden sonra bakım(maintenance) aşaması başlar. Sistemin donanım öğelerini bakım tutumu, temizleme, eskime ya da arızalanmış paçaların değiştirilmesi şeklinde yapılır veya tüm donanımın öğeleri yenisi ile değiştirilir. Yazılım öğelerinde bu eskime ya da 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 donanım bakımı gibi periyodik olmayıp, gelişen koşullara göre ortaya çıkar. Örneğin, daha önce hazırlanmış bir yazılımın 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 geliştirilmeli ve ona uyulmalıdır. IEEE 1219 Yazılım Bakım Standardı, bu konuda uygulanabilecek yöntemleri, araç ve teknikleri anlatmaktadır. Yazılım bakımı ile ilgili planlar ve içerikleri de IEEE/EIA 12207 Standardı içinde yer almaktadır. Bu bölümde yazılım bakımının nasıl yapılacağı hakkında temel bilgiler verilecektir.



7.2 Kurulum

Sınanmış yazılımların kullanıcı sahasına aktarılması ve yazılımın gerçek yaşamda uygulanmasının başlatılması için yapılan işlemler kurulum işlemleri olarak tanımlanmaktadır. Bütün bu kurulum işlemlerinin, dikkatlice planlanarak birbiri ile eşgüdümlü olarak yapılması gerekir. Özellikle büyük boyutlu, uzun zaman alan ve coğrafik olarak çok alana yayılmış projelerde eşgüdüm daha da önem kazanmaktadır.

Donanım kurulumu, yazılımın yerleştirileceği sahadaki donanımın, ağ alt yapısının ve kullanıcı uçlarının hazırlanma çalışmalarını içerir.

Sistem yazılımı kurulumu, işletim sistemi, hazır yordamlar, anti-virüs yazılımları ve kullanıcı tarafından kullanılacak ofis yazılımları türündeki yazılımların kurulumlarını içermektedir.

Veri tabanı kurulumu, uygulama yazılımları tarafından kullanılacak veri tabanı yazılımının (Oracle gibi) sunucu bilgisayarlara ve uygulamaya ilişkin veri tabanı platformunun istemci bilgisayarlara yüklenmesi işlemlerini içermektedir.

Uygulama yazılımlarının ana sunucu ve istemci sunulara yüklenmesi kurulum işleminin hedef işlemidir.

Kurulum işlemlerine koşut olarak kullanıcılara yazılım sistemlerine ilişkin eğitim verilmelidir. Söz konusu eğitim; uygulama yazılımı bakım eğitimidir.

Sistemlerin kurulumu sırasında ve sonrasında kullanıcı alanında sürekli kalınarak verilen destek “yerinde destek” olarak tanımlanmaktadır. İzleyen kesimde yerinde destek organizasyonunun yapısı açıklanmaktadır. Yerinde destek ekibi proje ekibinden seçilen bir ekip olacaktır. Planlama aşamasında eğitim, yazılım kurulumu vs. konulara ayrıntılı olarak değinilmişti.

7.3 Yerinde Destek Organizasyonu

Proje ekibi içinden seçilen bir ekip olup firma içinde yerinde destek amacıyla sözleşmeli personel olarak kalacaktır.

7.4 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.

Yazılımın bakımı ve onarımı (software maintenance) deyimi 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, sınaama aşamasında bulunamamış olan çeşitli işlem, yetenek ve tasarım hataları ortaya çıkabilmektedir. Bu hatalar; işlem sırasında kilitlenme, kesilme, yavaş çalışma, anormal işleme vb. olaylarla kendini göstermektedir. Ayrıca, yazılımın yeteneğini iyileştirmek ve kullanımını kolaylaştırmak üzere programa bazı eklemeler gerekebilmektedir. Donanımın, işletim sisteminin ya da verinin değiştirilmesi durumlarında da yazılımın yeni ortama uydurulması yoluna gidilmektedir. Bu durumda birkaç modül üzerinde değişiklik yapmak gerekmektedir. Ayrıca güncel olarak yeterli bulunmayan bir yazılımın; gereksinim analizi, tasarım, tamamlama ve sınaama basamakları olmak üzere yeniden geliştirilmesi ve böylece onarılması da söz konusudur. Yazılımın bakımı konusundaki işlerin %21’nin hata düzeltme, %25’inin iyileştirme, %50’sinin uyarılma ve %4’ünün diğer durumlarda olduğu bildirilmektedir.

Yazılımın bakımı ve korunması işlemi, kullanıma hazır oluşundan itibaren başlatılmakta ve kullanımdan kaldırılncaya kadar sürmektedir. Önce yazılım konfigürasyonu (software configuration) oluşturularak gerekli düzeltme ya da değişiklikler yerine getirilmektedir.

Bakım Maliyetlerinin Azaltılması

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

- 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 düzenleme kolaylıkları sağlamak

Bakım ve Onarıma Elverişlilik

Yazılımın bakım ve onarıma elverişliliğini ölçmek, diğer kalite faktörlerinde olduğu gibi ancak dolaylı olarak ve çok sayıda başka özelliklere dayanarak gerçekleştirilebilmektedir. Bu özellikler;

- Sorunun tanımı
- Yönetiminde gecikme
- Gerekli araçların derlenmesi
- Sorunun analizi
- Spesifikasyonlarının değiştirilmesi
- Düzeltme
- Sınama
- Gözden geçirme süreleri

olarak alınabilir. Bu bilgiler, yöneticiye de yeni teknikler ve araçlar kullanma ihtiyacını göstermektedir.

Yazılımın bakım ve onarıma elverişliliği; yazılımın diğer kalite faktörlerinde olan,

- Sınama kolaylığı
- Basitlik
- Değiştirilebilirlik
- Taşınabilirlik
- Güvenirlik
- Esneklik

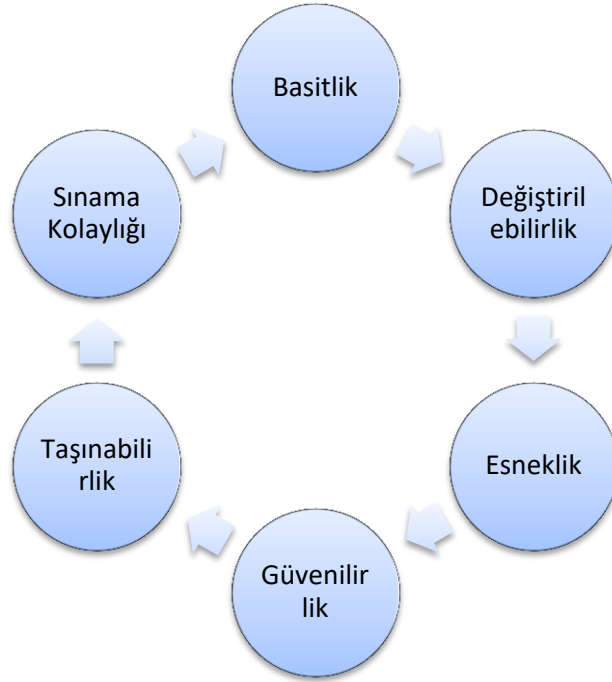
Özelliklerinin bir bileşkesi olarak ortaya çıkmaktadır. Bu nedenle ölçümünde bu faktörlerin varlığını denetlemek yolu da kullanılmaktadır.

Yazılım Bakımı ve Onarımına Elverişliliğin Önemi

Yazılımın bakım ve onarıma elverişliliği her 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, modülerite ve işlevsel bağımsızlık bakımlarından gözden geçirilmelidir. Kaynak programın dayanıklılığı ve program giriş belgeleri incelenmelidir. Sınama konfigürasyonu gözden geçirilmelidir (configuration review). Burada; yazılım düzeninin bütün öğelerinin tamam,

anlaşılabilir ve bakım-onarım denetiminde uygun biçimde kütüklere ayrılmış bulunduğu saptanmaktadır.

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



Yazılım Bakımı

- Tanım
- Bakım Süreç Modeli
 - Sorun Tanımlama Süreci
 - Çözümleme Süreci
 - Tasarım Süreci
 - Gerçekleştirim Süreci
 - Sistem Sınama Süreci
 - Kabul Sınama Süreci
 - Kurulum Süreci

7.4.1 Tanım

Yazılım bakımı, müşteri ile geliştirici arasında yapılan sözleşmeye göre bazen yazılım geliştirme sürecinde yer almamaktadır. Oysa günümüzdeki pek çok uygulama alanı değişen ihtiyaçlara sahiptir. Bu nedenle yazılım geliştirilip kullanıma sunulduktan sonrasında da 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 temel olarak alınmıştır.

- **Düzenleyici Bakım:** Yazılım testi her zaman tüm kusurların bulunmasını ve giderilmesini sağlamaz. Çalışan bir yazılımda her an hata olasılığı vardır. Bir kısım yazılım hataları ancak kullanım sırasında ortaya çıkar. Bu hatalar giderilmesi için geliştiriciye bildirilir. Hatanın sebebini araştırmaya ve gidermeye yönelik işlere düzeltici (corrective) bakım adı verilir. Kullanım sırasında bulunan ve raporlanan yazılım hatalarının giderilmesi, hatanın önemine bağlı olarak ya hemen düzeltilmesi yapılır ya da birkaç tanesi ile beraber düzeltilmesi yapılır ve yeni bir sürüm çıkarılır. Bu bakım, geliştiricinin bakım aşamasında mutlaka yürütmesi gereken bir etkinliktir.
- **Uyarlayıcı Bakım:** Bilgi işleme dünyasındaki hızlı değişimler ve teknolojik gelişmeler nedeniyle yazılımın yeni donanıma, işletim sistemlerine, bunların yeni sürümlerine, yeni uçbirimlere göre uyarlanması, sürümün yükseltilmesi ve güncelleştirilmesi işlemlerine uyarlayıcı (adaptive) bakım denmektedir.
- **İyileştirici Bakım:** Yazılım geliştirilip test edilerek 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 sürümleri ortaya çıkar ve kullanıcının hizmetine sunulur.
- **Önleyici Bakım:** Yazılımın, gelecekte uygulanabilecek değişikliklere daha iyi bir temel oluşturması, bakıla bilirliğinin ve güvenilirliğinin artırılması için ön tedbir niteliğindeki işlemler önleyici (preventive) bakım kapsamına girmektedir.

Yazılım bakımı, müşteri ile geliştirici arasında yapılan sözleşmeye göre bazen yazılım geliştirme sürecinde yer almamaktadır. Oysa günümüzdeki pek çok uygulama alanı değişen ihtiyaçlara sahiptir. Bu nedenle yazılım geliştirilip kullanıma sunulduktan sonrasında da 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 temel olarak alınmıştır.

- **Düzenleyici Bakım:** Yazılım testi her zaman tüm kusurların bulunmasını ve giderilmesini sağlamaz. Çalışan bir yazılımda her an hata olasılığı vardır. Bir kısım yazılım hataları ancak kullanım sırasında ortaya çıkar. Bu hatalar giderilmesi için geliştiriciye bildirilir. Hatanın sebebini araştırmaya ve gidermeye yönelik işlere düzeltici (corrective) bakım adı verilir. Kullanım sırasında bulunan ve raporlanan yazılım hatalarının giderilmesi, hatanın önemine bağlı olarak ya hemen düzeltilmesi yapılır ya da birkaç tanesi ile beraber düzeltilmesi yapılır ve yeni bir sürüm çıkarılır. Bu bakım, geliştiricinin bakım aşamasında mutlaka yürütmesi gereken bir etkinliktir.
- **Uyarlayıcı Bakım:** Bilgi işleme dünyasındaki hızlı değişimler ve teknolojik gelişmeler nedeniyle yazılımın yeni donanıma, işletim sistemlerine, bunların yeni sürümlerine, yeni uçbirimlere göre uyarlanması, sürümün yükseltilmesi ve güncelleştirilmesi işlemlerine uyarlayıcı (adaptive) bakım denmektedir.
- **İyileştirici Bakım:** Yazılım geliştirilip test edilerek başarılı bir şekilde kullanıcıya sunulduktan sonra yeni işlevler eklenerek, var olanlara başarımlı ve verimi artırıcı düzeltmeler yapmak iyileştirici (perfective) bakım işleri içinde yer alır. Bu şekilde yazılımın yeni sürümleri ortaya çıkar ve kullanıcının hizmetine sunulur.
- **Önleyici Bakım:** Yazılımın, gelecekte uygulanabilecek değişikliklere daha iyi bir temel oluşturması, bakıla bilirliğinin ve güvenilirliğinin artırılması için ön tedbir niteliğindeki işlemler önleyici (preventive) bakım kapsamına girmektedir.

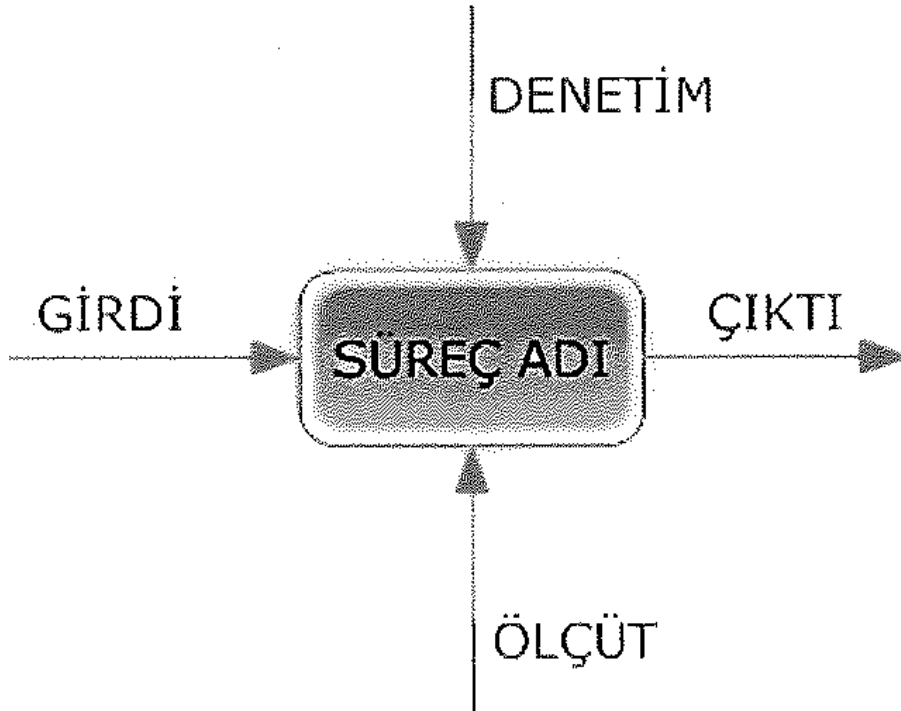
7.4.2 Bakım Süreç Modeli

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

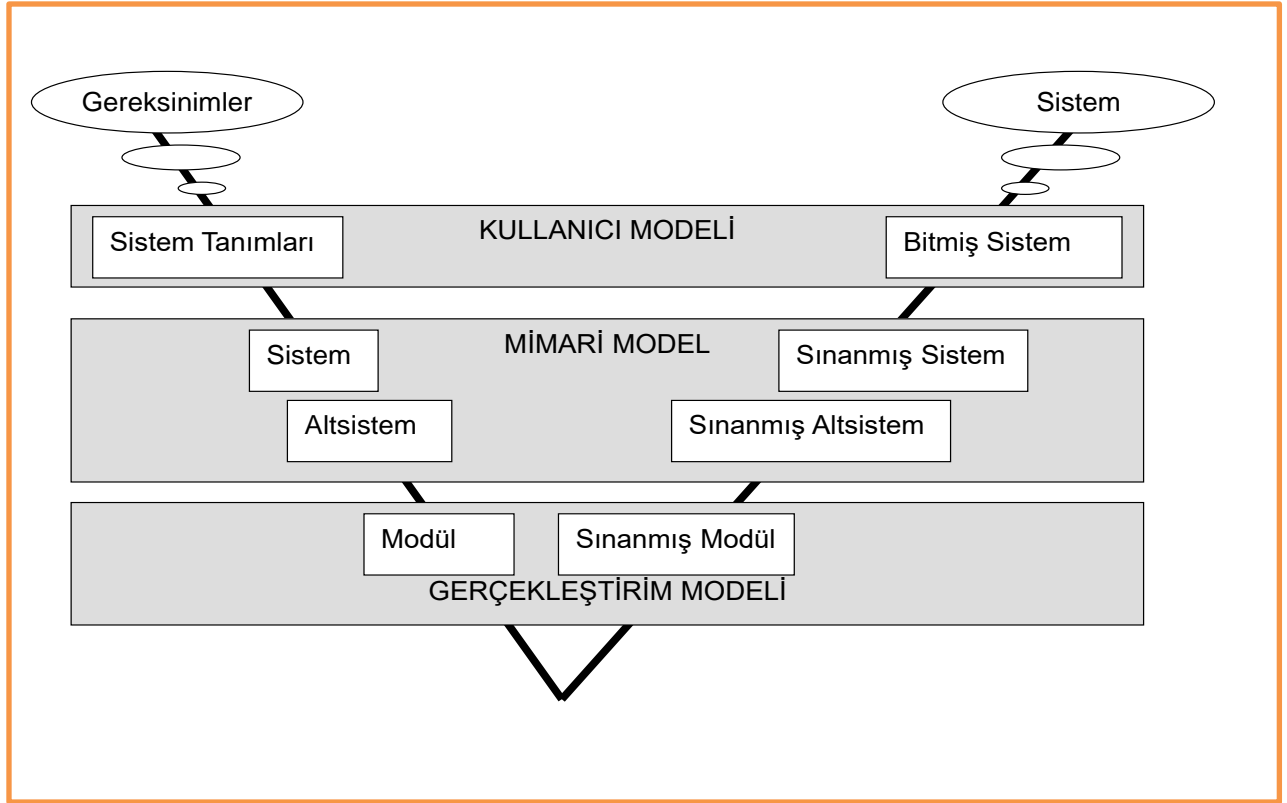
Bakım süreç modelinin süreçleri ;

- 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 yinelenme yalnızca değişiklik isteklerinin var olan koda aktarılması amacıyla yapılmaktadır. Bakım süreçleri aşağıda açıklanmaktadır.



Projede bakım geliştirme modeli olarak “V Süreç Modeli” kullanımı tercih edilmiştir.

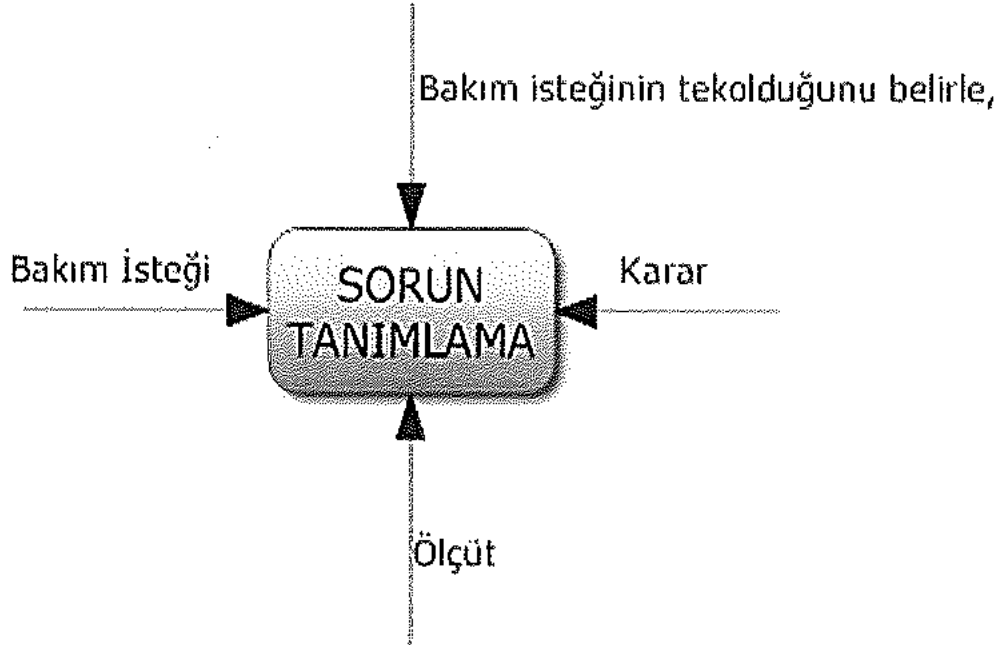


- Sol taraf üretim, sağ taraf sınama işlemleridir.
- V süreci modelinin temel çıktıları;
 - Kullanıcı Modeli → Geliştirme sürecinin kullanıcı ile olan ilişkileri tanımlanmakta ve sistemin nasıl kabul edileceğine ilişkin sınama belirtileri ve planları ortaya çıkarılmaktadır.
 - Mimari Model → Sistem tasarımı ve oluşacak alt sistem ile tüm sınama işlemlerine ilişkin işlevler.
 - Gerçekleştirim Modeli → Yazılım modüllerinin kodlaması ve sınamasına ilişkin fonksiyonlar
- Belirsizliklerin az iş tanımlarının belirgin olduğu projeler için uygun bir modeldir.
- Model, kullanıcının projeye katkısını artırmaktadır.
- Projenin iki aşamalı olarak ihale edilmesi oldukça uygundur;
 - İlk ihalede kullanıcı modeli hedeflenerek iş analizi ve kabul sınamalarının tanımları yapılmaktadır
 - İkinci ihalede ise ilkinde elde edilmiş olan kullanıcı modeli tasarlanıp, gerçekleştirilmektedir.

Bu modelde adından da anlaşılacağı gibi “V” yapısında bir yol izlenir ve adımlar bu şekilde gerçekleştirilir. Bu yol üzerinde sol taraf üretimi, sağ taraf ise test işlemini ifade eder. Bu modelde yer alan çıktıları “Kullanıcı Modeli”, “Mimari Model” ve “Gerçekleştirim Modeli” adı altında toplayabiliriz. Kullanıcı modelinde, geliştirme sürecinin kullanıcı ile olan ilişkileri tanımlanmakta ve sistemin nasıl kabul edileceğine ilişkin sınama belirtileri ve planları ortaya çıkarılmaktadır. Mimari modelde, sistem tasarımı ve oluşacak alt sistem ile tüm

sitemin sınaıa işlemlerine ilişkin işlevler ele alınmaktadır. Gerçekleştirim modelinde de yazılım modüllerinin kodlanması ve sınanmasına ilişkin fonksiyonlar ele alınmaktadır. Bu model belirsizliklerinin az iş tanımlarının belirgin olduğu bilirim teknolojileri projeleri için uygun bir modeldir. Ayrıca model kullanıcının projeye katkısını arttırmaktadır.

A . Sorun Tanımlama Süreci:



Girdi

Sorun tanımla sürecinin temel girdisi, Bakım isteğidir. Herhangi bir bakım isteğine örnek olarak; Sistemde beklenen ve yeni düzenlemeler ilişkin değişiklikler,

- Yeni işlev istekleri
- Yazılımda bulunan yanlışlıkların düzeltilme istekleri
- Performans artırımına ilişkin istekler
- Yeni iş yapma türlerine ilişkin istekler
- Teknolojinin zorlanması sonucu oluşan istekler

türündeki istekler verilebilir.

İşlem / Sürec

Yazılım bakım isteği oluştuğunda yapılması gereken işlemler;

- Değişiklik isteğine bir tanım numarası verilmesi
- Değişiklik türünün sınıflandırılması
- Değişiklik isteğinin kabul, ret ya da daha ayrıntılı incelenmesi yönünde karar alınması
- Değişiklik isteğinin önceliklendirilmesi

- Değişiklik isteğinin diğerleri ile birlikte zaman ve iş planına kaydedilmesi

biçiminde özetlenebilir. Bu işlemlerin birçoğunda değişiklik isteğini yapan kişi, kullanıcı temsilcileri, yazılım mühendisleri ve konu uzmanları birlikte çalışıp karar verirler.

Denetim

Sorun tanımlama aşamasında değişiklik isteğinin daha önceden yapılıp yapılmadığı denetlenmeli ve tek olduğu belirlenmelidir. Bu amaçla daha önceki değişiklik istekleri taranır.

Çıktı

Bu sürecin temel çıktısı; doğrulanmış, geçerlenmiş ve karar verilmiş Bakım İsteğidir. Bir veri tabanında saklanan bu isteğin ayrıntıları;

- Sorun ya da yeni gereksinimin tanımı
- Sorun ya da gereksinimin değerlendirilmesi
- Başlangıç öncelik
- Geçerleme verisi (Düzeltilici bakım için)
- Başlangıç kaynak gereksinimi
- Mevcut ve gelecekte kullanıcılar üzerindeki etkileri
- Yararlı ve aksak yönler

Ölçüt

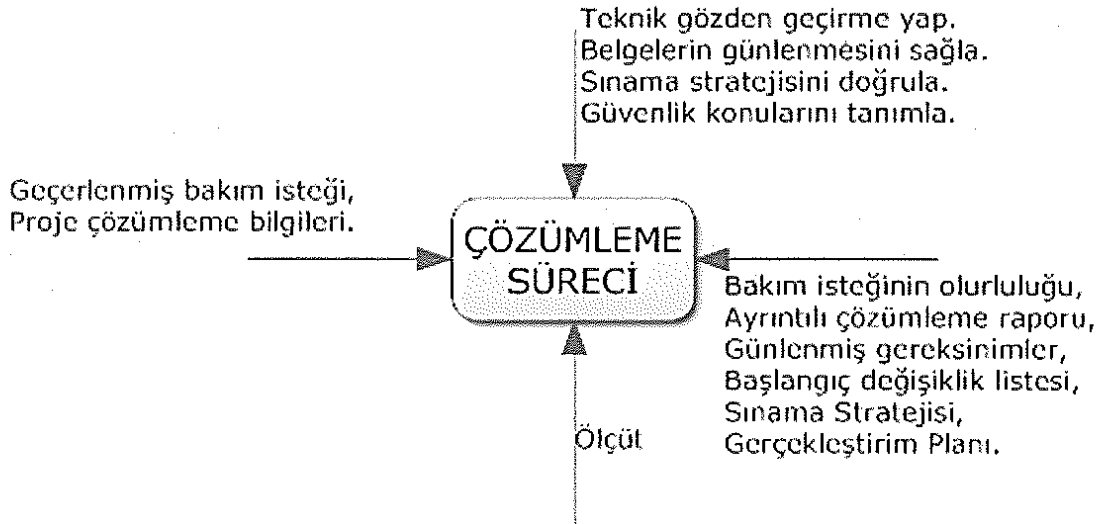
Sorun tanımlama sırasında kullanılabilecek ölçütler;

- Bakım isteklerinde kabul edilmeyen madde sayısı
- Gelen bakım istekleri sayısı
- Sorun geçerleme için harcanan kaynak ve zaman

biçimindedir.

B . Çözümleme Süreci :

Veri tabanında saklanmış ve geçeklenmiş bakım isteğini girdi olarak alır, projeye ilişkin bilgi ve belgeleri kullanarak söz konusu isteğin yerine getirilmesi için gerekli genel planı yapar.



Girdi

Çözümleme sürecinin girdileri;

- Geçerlenmiş bakım isteği
- Başlangıç kaynak gereksinimleri ve diğer veriler
- Mevcut proje ya da sistem bilgi ve belgeleri

biçimindedir.

İşlem / Sürec

Çözümleme süreci temel olarak iki aşamadan oluşur. Olurluk aşaması ve ayrıntılı çözümleme aşaması. Mevcut sisteme ya da projeye ilişkin yapısal belgelerin bulunmadığı durumlarda tersine mühendislik yöntemi kullanılmıştır.

Olurluk çalışması;

- Değişikliğin etkisi
- Prototiplemeyi içeren seçenek çözümler
- Dönüştürme gereksinimlerinin çözümlemesi
- Güvenlik ve emniyet zorunlulukları
- İnsan faktörleri
- Kısa ve uzun erimli maliyetler
- Değişikliği yapmanın yararları

bilgilerini içerir.

Ayrıntılı çözümleme çalışmasında, değişiklik isteği için ayrıntılı gereksinim tanımlaması yapılır. Bu çalışmada etkilenen yazılım öğeleri (yazılım tanımları, yazılım gereksinimleri, tasarım, kod, vb.) belirlenir. Yazılım öğelerinin değişmesi gereken kısımları belirlenir. En az üç düzeyli sınama stratejisi (birim sınaması, bütünleştirme sınaması ve kabul sınaması)

tanımlanır. Bu aşamanın son çalışması ise kullanıcıya en az etki yapacak şekilde değişiklik gereksinimlerinin nasıl karşılanacağı bilgilerini içeren “Başlangıç Gerçekleştirim Planı”dır.

Denetim

Çözümleme çalışmasının denetiminde aşağıdaki işlemler yapılır;

- Gerekli proje ya da sistem bilgi/belgelerine erişimin sağlanması (ortam denetim organizasyonundan)
- Önerilen değişikliklerin ve çözümleme çalışmasının teknik ve ekonomik olurluğunun gözden geçirilmesi
- Güvenlik ve emniyet konularının tanımlanması
- Önerilen değişikliğin, mevcut yazılımla bütünleştirilmesinin dikkate alınması
- Proje belgelerinin düzgün olarak günlendiğinin denetimi
- Çözümleme belgelerinin düzgün olarak hazırlanmasının sağlanması
- Sınama stratejilerinin uygun olarak belirlenmesi

Çıktı

Çözümleme çalışmasının çıktıları şunlardır;

- Değişiklik isteklerine ilişkin olurluk çalışması
- Ayrıntılı çözümleme raporu
- İzlenebilirlik listesini içeren güncellenmiş gereksinim tanımları
- Başlangıç değişiklik listesi
- Sınama stratejisi
- Gerçekleştirim planı

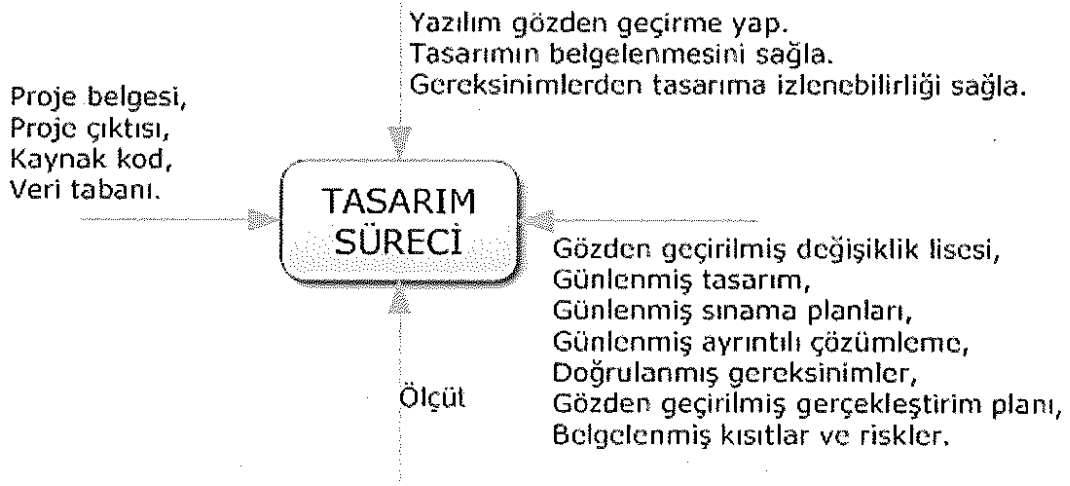
Ölçüt

Çözümleme çalışmasında kullanılabilecek ölçütler şu biçimdedir;

- Gereksinimlerdeki değişiklik sayısı
- Belgeleme hata oranı
- Her işlev alanı için gerekecek işgücü
- Toplam zaman

C . Tasarım Süreci:

Tasarım aşamasında, değişiklikten etkilenebilecek tüm proje bilgi ve belgeleri üzerinde çalışma yapıp söz konusu bilgi ve belgeler değişiklikle ilgili olarak güncellenir.



Girdi

Tasarım çalışmasının girdileri şu şekildedir;

- Çözümleme çalışması çıktıları
 - Ayrıntılı çözümleme
 - Günlenmiş gereksinim tanımları
 - Başlangıç değişiklik listesi
 - Sinama stratejisi
 - Gerçekleştirim planı
- Sistem ve proje belgeleri
- Var olan kaynak kodları açıklamaları ve veri tabanları

İşlem / Sürec

Tasarım için gerekli temel işlemler aşağıda belirtilmektedir;

- Etkilenen yazılım modüllerinin tanımlanması
- Yazılım modül belgelerinin değiştirilmesi
- Yeni tasarım için güvenlik ve emniyet konularını da içerek sinama senaryolarının hazırlanması
- İlişki sinamalarının tanımlanması
- Kullanıcı belgelerinin günleme gereksinimlerinin tanımlanması
- Değişiklik listesinin günlenmesi

Denetim

Tasarım çalışmasında aşağıdaki denetim ortamı kurulmalıdır. Tasarımın belirlenen standartlara uygunluğunun denetlenmesi;

- Güvenlik ve emniyet konularını içeren yeni tasarım belgesinin ve bilgilerinin oluşturulmasının sağlanması

- Sınama bilgilerinin g nlenmesinin saėlanması
- Gereksinimlerden tasarıma izlenebilirliėin saėlanması

Çıktı

Bakım tasarımı alıřmasının ıktıları řu řekildedir;

- G zden geirilmiş deėişiklik listesi
- G nlenmiř tasarım
- G nlenmiř sınama planları
- G nlenmiř ayrıntılı  z mlleme
- G nlenmiř gereksinimler
- G zden geirilmiş gerekleřtirim planı
- Risk ve kısıtlar listesi

 l t

Tasarım alıřması iin kullanılabilen  l tler ařaėıda verilmektedir;

- Yazılım karmařıklıėı
- Tasarım deėişiklikleri
- Her iřlev alanı iin gerekecek iřg c 
- Toplam zaman
- Sınama y nergeleri ve plan deėişiklikleri
-  nceliklendirmedeki hata oranları
- Var olan kodda eklenen, ıkarılan ve deėiřtirilen satır sayısı
- Uygulama sayısı

D . Gerekleřtirim S reci:

Temel olarak tasarım ıktılarını ve kaynak kodları girdi olarak almakta ve deėişiklik isteėini gerekleřtiren kod paraları ile g nlenmiř yazılım kodlarını  retmektedir. G nlenmiř yazılıma iliřkin sınama bilgi ve belgelerinin ve eėitim belgelerinin  retimi de bu s rete yapılmaktadır.



Girdi

Gerçekleştirim sürecinin girdileri şu şekildedir;

- Tasarım çalışması sonuçları
- Var olan kaynak kodlar, açıklamalar, belgeler
- Proje ve sistem belgeleri

İşlem / Sürec

Gerçekleştirim sürecinin dört ana işlemi vardır;

- Kodlama ve birim sınama
- Bütünleştirme ve risk çözümleme
- Sınama hazırlığı gözden geçirme
- Kodlama işleminde; değişiklik isteğini karşılayan yazılım kodları, var olan yazılıma eklenmektedir. İşlem sonucunda elde edilen yeni, değişmiş modüllere birim sınama uygulanmaktadır. Birim sınama işlemini, bütünleştirme-sınama izlemekte, tüm sistem yeniden sınanmaktadır. Uygulamadaki riskleri gidermek amacıyla gerçekleştirim aşamasında sürekli risk çözümleme yapılmaktadır

Denetim

Gerçekleştirim sürecinde oluşturulacak denetim yapısı aşağıdaki özellikleri sağlamalıdır;

- Belirlenen standartlara uygun olarak kod ve yazılım gözden geçirmeleri yapılması
- Birim ve bütünleştirme sınamaları ile ilgili bilgilerin derlenmesi ve kaydedilmesinin sağlanması
- Sınama belgelerinin güncellenmesi ve oluşturulmasının sağlanması
- Sınama hazırlık gözden geçirmeleri sırasında risk çözümlemenin sağlanması
- Yeni yazılımın, yazılım ortam yönetimi altında kaydedilmesi ve denetlenmesinin sağlanması
- Teknik ve eğitim belgelerinin güncellenmesinin sağlanması
- Tasarımdan koda izlenebilirğin sağlanması

Çıktı

Gerçekleştirim süreci aşağıdaki çıktıları vermelidir;

- Günlenmiş yazılım
- Günlenmiş tasarım bilgi7belgeleri
- Günlenmiş sınaıa belgeleri
- Günlenmiş kullanıcı belgeleri
- Günlenmiş eğitim kılavuzu
- Riskler ve kullanıcılara etkileri
- Sınama hazırlığı gözden geçirme, rapor

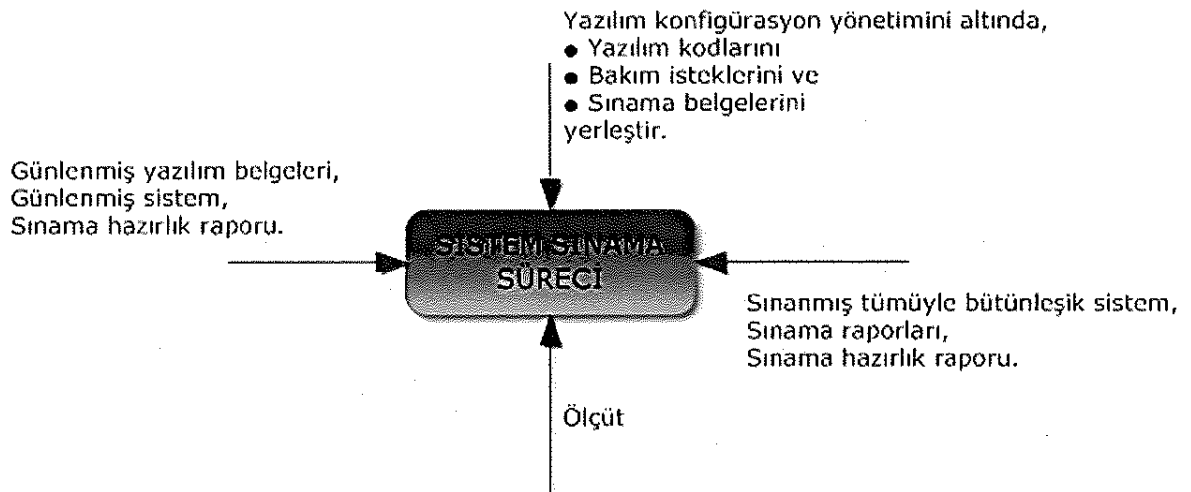
Ölçüt

Gerçekleştirim çalışmasında kullanılabilecek ölçütler şunlardır;

- Değişiklik oranı
- Hata oranı

E . Sistem Sınama Süreci :

Değişikliklerin var olan yazılıma yansıtılmasından sonra elde edilen yeni yazılım sürümünün belirlenen standartlara uygun olarak tümüyle bütünleşik sistem üzerine sınamaların yapılması gerekmektedir. Sistem sınamalarının, kullanıcı ve üretici ekiplerin tanıklığında bağımsız bir yapı tarafından gerçekleştirilmeleri önerilmektedir.



Girdi

Sistem sınaıa sürecinin girdileri şöyledir;

- Sınama hazırlık raporu
- Belgeler
 - Sistem sınaıa planları
 - Sistem sınaıaları
 - Sistem sınaıa yönergeleri
 - Kullanıcı kılavuzları
 - Tasarım
- Günlennmiş sistem

İşlem / Sürec

Sistem sınaıa, tümüyle bütünleşik bir sistem üzerinden yapılmalıdır. Bu aşamada işlevsel sistem, ara yüz sınaıa, regresyon sınaıa ve sınaıa hazırlık raporunun gözden geçirilmesi işlemleri yapılır.

Denetim

Sistem sınaıaları, üretici ve kullanıcılardan bağımsız bir grup tarafından gerçekleştirilmelidir. Yazılım kodları ve her türlü bilgi, belge, yazılım ortam yönetimi tarafından saklanır.

Sistem sınaıa sürecinde oluşacak denetim yapısı aşağıdaki özellikleri sağlamalıdır;

- Belirlenen standartlara uygun olarak kod ve yazılım gözden geçirmeleri yapılması
- Birim ve bütünleştirme sınaıaları ile ilgili bilgilerin derlenmesi ve kaydedilmesinin sağlanması
- Sınama belgelerinin günlenmesi ve oluşturulmasının sağlanması
- Sınama hazırlık gözden geçirmeleri sırasında risk çözümlemenin saplanması
- Yeni yazılımın, yazılım ortam yönetimi altına kaydedilmesi ve denetlenmesinin sağlanması
- Teknik ve eğitim belgelerinin günlenmesinin sağlanması
- Tasarımdan koda izlenebilirliğin sağlanması

Çıktı

Sistem sınaıa süreci aşağıdaki çıktıları vermelidir;

- Günlennmiş yazılım
- Günlennmiş tasarım bilgi/belgeleri
- Günlennmiş sınaıa belgeleri

- Günlenmiş kullanıcı belgeleri
- Günlenmiş eğitim kılavuzları
- Riskler ve kullanıcılara etkileri
- Sınama hazırlığı gözden geçirme, rapor

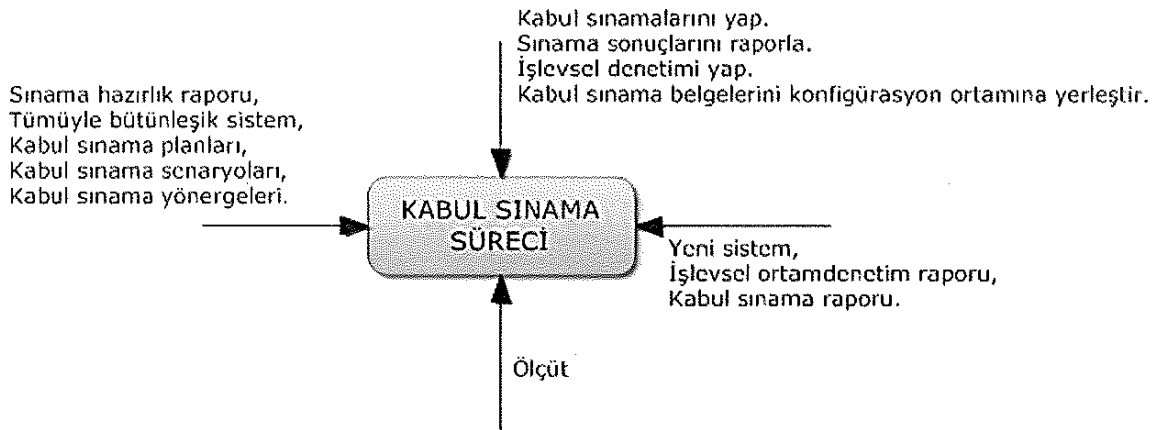
Ölçüt

Sistem sınama çalışmasında kullanılabilecek ölçütler şunlardır;

- Değişiklik oranı
- Hata oranı

F . Kabul Sınama Süreci :

Kullanıcılar ya da kullanıcı temsilcileri tarafından gerçekleştirilen bir süreçtir. Kullanıcıların, değişiklikleri içeren yeni yazılım sınamaları ve kabul etmeleri beklenmektedir.



Girdi

Kabul sınama sürecinin girdileri şöyledir;

- Gözden geçirilmiş sınama hazırlık raporu
- Tümüyle bütünleşik sistem
- Kabul sınama planları
- Kabul sınamaları
- Kabul sınama yönergeleri

İşlem / Süreç

Kabul sınama işlemleri şöyledir;

- İşlevsel kabul sınamalarının yapılması
- Birlikte çalışabilirlik sınaması
- Regresyon sınaması

Denetim

Kabul sınamaları sırasında denetimi aşağıdaki işlemleri içermektedir;

- Kabul sınamalarının uygulanması
- Sınama sonuçlarının raporlanmasının sağlanması
- İşlevsel denetim yapılması
- Yeni sistem oluşturulması
- Kabul sınama belgelerinin yazılım konfigürasyonuna yerleştirilmesi

Çıktı

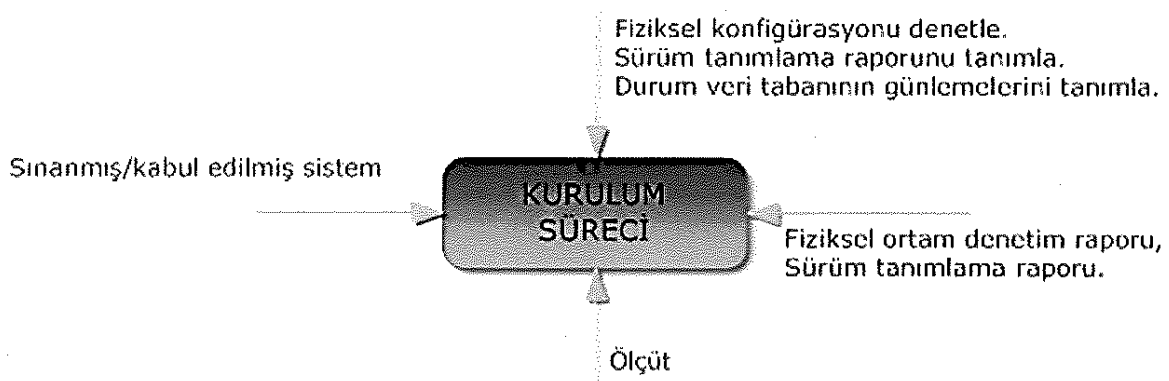
Kabul sınamaları çıktıları; yeni sistem, işlevsel konfigürasyon denetim raporu ve kabul sınaması raporudur.

Ölçüt

Bu aşamada kullanılabilecek ölçütler; üretilen ve düzeltilen hata oranlarıdır.

G . Kurulum Süreci :

Geliştirilen ya da değiştirilmiş yeni yazılım sürümünün, uygulama sahasına aktarılması işlemlerini içerir.



Girdi

Bu sürecin temel girdisi; tümüyle sınanmış ve kabul edilmiş yeni yazılım sürümüdür.

İşlem / Sürec

Bu sürecim işlemleri şöyledir;

- Fiziksel ortam denetiminin yapılması
- Kullanıcıların bilgilendirilmesi
- Var olan sistem yedeklerinin alınması
- Kullanıcı tarafından kurulum ve eğitimlerin yapılması

Denetim

Denetim işlemleri şu alanları içermektedir;

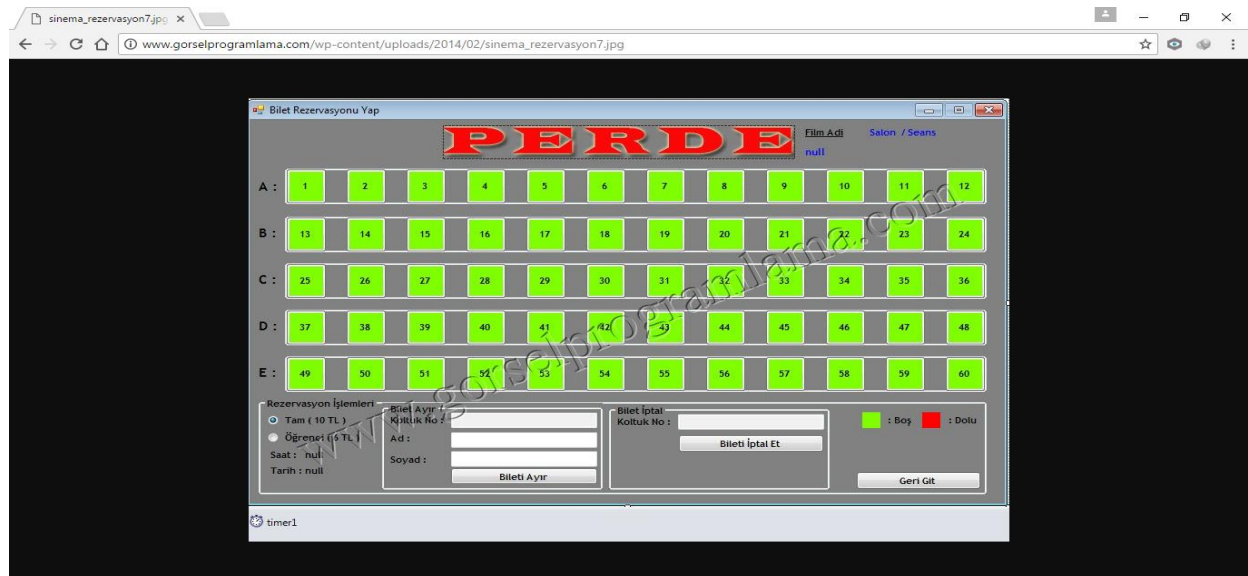
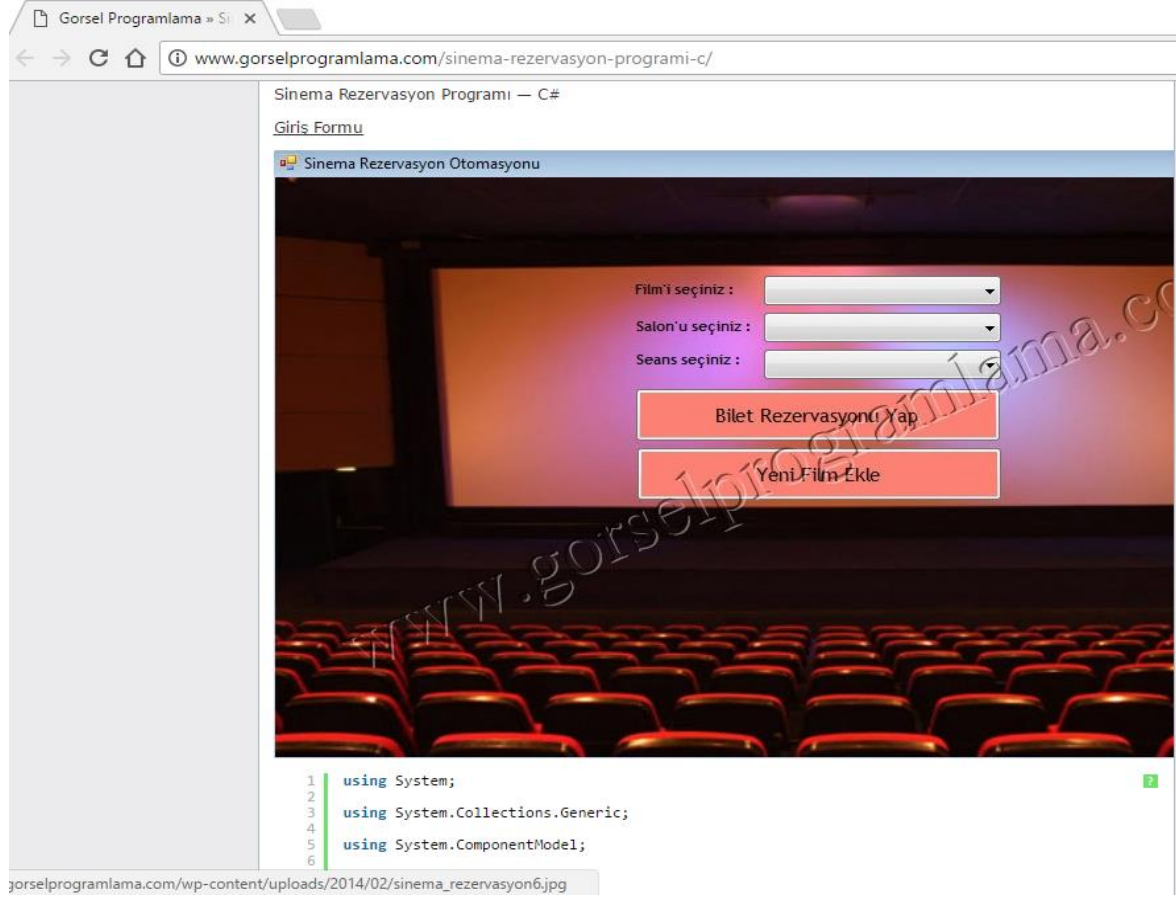
1. Fiziksel ortam denetiminin yapılması
2. Sistem ile ilgili bilgi ve belgelerin kullanıcıya ulaştırılması
3. Sürüm Tanımlama raporunun tamamlanması
4. Yazılım konfigürasyon ortamına aktarımının sağlanması

Çıktı

5. Bu sürecin temel çıktıları; fiziksel ortam denetim raporu ve Sürüm Tanımlama raporudur.
6. **Ölçüt**
7. Bu süreçte kullanılabilecek ölçüt, belgeleme değişiklikleridir.

8. Kaynakça

- 1) <http://www.gorselprogramlama.com/sinema-rezervasyon-programi-c/>



- 2) Ali Arifoğlu' nun Hazırlamış olduğu Yazılım Mühendisliği Temelleri Slaytı
- 3) Yazılım Mühendisliği Temelleri Slaytları
- 4) Servet Öksüz'ün 2014-2015 yılları arasına yazmış olduğu 'İnsan Bağımsız Güvenlik Sistemli Online Oyun' Projesi
- 5) Atilla Özgür'ün yazmış olduğu 'Sosyal Yardımlaşma ve Dayanışma Otomasyon Sistemi' Projesi
- 6) <http://tr.wikipedia.org/wiki/Gantt%C3%A7izelgesi>
- 7) https://tr.wikipedia.org/wiki/Spiral_model
- 8) <http://muhammetbaykara.com/dersler/yazilim-muhendisliginin-temelleri/>
- 9) <http://muhammetbaykara.com/wp-content/uploads/2016/11/k%C4%B1s%C4%B1m1.pdf>
- 10) <http://muhammetbaykara.com/wp-content/uploads/2016/11/ks%C4%B1m2.pdf>
- 11) <http://muhammetbaykara.com/wp-content/uploads/2016/11/k%C4%B1s%C4%B1m3.pdf>
- 12) <http://muhammetbaykara.com/wp-content/uploads/2016/11/k%C4%B1s%C4%B1m4.pdf>
- 13) <http://muhammetbaykara.com/wp-content/uploads/2016/11/k%C4%B1s%C4%B1m5.pdf>
- 14) <http://muhammetbaykara.com/wp-content/uploads/2016/11/k%C4%B1s%C4%B1m6.pdf>
- 15) <http://muhammetbaykara.com/wp-content/uploads/2017/02/ymh-114-proje-%C5%9Fablonu.pdf>
- 16) <http://muhammetbaykara.com/wp-content/uploads/2017/02/%C3%96dev-ve-Proje-De%C4%9Ferlendirme-%C3%96l%C3%A7%C3%BCtleri.pdf>
- 17) Form Geçişleri hakkında videolar
- 18) Kullanıcı Otomasyonu hakkında video
- 19) OleDbVeritabanı Hakkında örnekler
- 20) RehberV3 uygulaması