T.C.
FIRAT ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

Proje Dokümantasyonu

Tez Kuralları Kontrol Projesi

Proje Ekibi

Recai Çapkın 16541047

Melahat Erbaş 16541008

2020 – 1.0

1. GİRİŞ

1.1 Projenin Amacı

1.2 Projenin Kapsamı

1.3 Tanımlamalar ve Kısaltmalar

2. PROJE PLANI

2.1 Giriş

2.2 Projenin Plan Kapsamı

2.3 Proje Zaman-İş Planı

2.4 Proje Ekip Yapısı

2.5 Önerilen Sistemin Teknik Tanımları

2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları

2.7 Proje Standartları, Yöntem ve Metodolojiler

2.8 Kalite Sağlama Planı

2.9 Konfigürasyon Yönetim Planı

2.10 Kaynak Yönetim Planı

2.11 Eğitim Planı

2.12 Test Planı

2.13 Bakım Planı

2.14 Projede Kullanılan Yazılım/Donanım Araçlar

3. SİSTEM ÇÖZÜMLEME

3.1 Mevcut Sistem İncelemesi

3.1.1 Örgüt Yapısı

3.1.2 İşlevsel Model

3.1.3 Veri Modeli

3.1.4 Varolan Yazılım/Donanım Kaynakları

3.1.5 Varolan Sistemin Değerlendirilmesi

3.2 Gereksenen Sistemin Mantıksal Modeli

3.2.1 Giriş

3.2.2 İşlevsel Model

3.2.3 Genel Bakış

3.2.4 Bilgi Sistemleri/Nesneler

3.2.5 Veri Modeli

3.2.6 Veri Sözlüğü

3.2.7 İşlevlerin Sıradüzeni

5. SİSTEM GERÇEKLEŞTİRİMİ

1. Giriş

1.1 Projenin Amacı

Projenin temel amacı verilen bir tez metini içerisinde yazım hatalarının belirlenmesi ve kurallara uygunluklarının denetlenmesi amaçlanmaktadır.

1.2 Projenin Kapsamı

Akademik makale yazmak isteyen her kişinin çalışmasını belirlenen standartlara uygun hale getirebilmek adına yapılan hataların tespit edilmesi ve raporlanması amacı ile kullanılacaktır.

1.3 Tanımlamalar Ve Kısaltmalar

(Regex): rx
(Lexeme): lx
(Tokenize): tkz
(Token): tk
(Lexical Analiz): la
(Portable Document Format): PDF

2. Proje Planı

2.1 Giriş

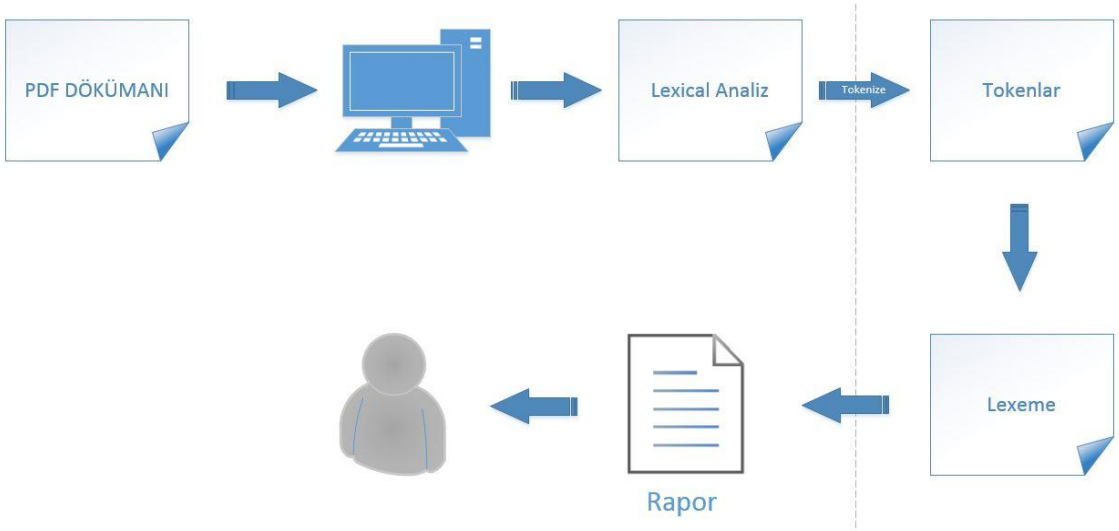
Uygulamanın amacı, kullanıcıdan girdi olarak alınan pdf dokümanının belirlenen standartlara uygunluğunun rx ifadelerinin kullanılarak la ile yapılan tkz işleminin daha anlamlı bir hale getirilebilmesi için tk ‘lara ayrılması ve bu ayırma işlemi sonucunda lx ‘lerin gruplandırılarak kontrol edilmesi, kullanıcıya raporlanmasıdır.

Şekil 2.1.1 Bu şekilde lx yapısının temel çalışma mantığı gösterilmiştir.

Şekil 2.1.2 Verilen bu şekilde girdi olarak alınan bir program tanımının metinsel sözdizimi analizi yapılmıştır.

2.2 Projenin Plan Kapsamı

Pdf dokümanı uygulamaya girdi olarak verilir. Uygulama verilen girdide la ile tkz işlemi gerçekleştirildi. Elde edilen tk ‘lar ayrılarak lx ‘ler elde edildi. Elde edilen lx ‘lerin standartlara uygunluğu kontrol edilerek rapor haline getirildi ve kullanıcıya sunuldu.



Şekil 2.2.1 Projenin Genel Plan Kapsamı

2.3 Proje Zaman-İş Planı

Şekil 2.3.1 Zaman İş Planı Gantt Diyagramı

2.4 Proje Ekip Yapısı

Şekil 2.4.1 Ekip Yapısı

"Proje teslim klasörü içerisinde yer alan tüm bilgi, belge ve dokümanlar tarafımdan gözden geçirilmiştir. Ekip yapısı içerisinde belirtilen görev ve sorumluluklardan tarafıma atanan işleri titizlikle gerçekleştirdiğimi beyan ederim. Puanlandırma bilgim ve aktif katılımım ile belirlenmiştir. Alınan nihai karar oy çokluğu ile alınması durumunda ekip üyeleri tarafından yapılan itirazlar ek olarak sunulmuştur ve bilgim dahilindedir."

Melahat ERBAŞ -16541008 -Onaylanmıştır.
Recai Çapkın -16541047 -Onaylanmıştır.

Proje Ekibi Değerlendirme Metriği

1. **Yüksek Başarılı(5 Puan)**
2. **Başarılı(4 Puan)**
3. **Orta Başarılı(3 Puan)**
4. **Düşük Başarılı(2 Puan)**
5. **Başarısız(1 Puan)**

Melahat Erbaş Değerlendirme:

Sistem Yöneticisi(Yüksek Başarılı): İhtiyaç analizleri gerçekleştirilirken olması gereken özellikleri dikkatli bir şekilde incelemiştir ve rapor etmiştir.

Sistem Tasarımcısı(Yüksek Başarılı): Programlama esnasında sistemin en performanslı çalışması için gerekli kodları yazım anında düzenlemiştir ve denetlemiştir.

Veri Tabanı Yöneticisi(Orta Başarılı): Projemiz için veri tabanı kullanımına gerek görülmemiştir fakat doküman için gerekli raporlamalar yapıldığın ötürü bu metrik ile değerlendirilmiştir.

UI Tasarımcı(Yüksek Başarılı): Projenin en kullanışlı ve en kolay şekilde ulaşılabilir komponentlere sahip olmasında büyük emeği geçmiştir.

Bu değerlendirmeler Recai Çapkın tarafından tarafsız bir şekilde yapılmıştır.

Recai Çapkın Değerlendirme:

Proje Yöneticisi (Yüksek Başarılı): Yazılacak modüllerin ve ara yüzlerin zorluk dereceleri tespit ederek zaman tayinini verimli bir şekilde değerlendirmiştir. Bilgi alışverişini pozitif yönde etkileyerek gerekli motivasyonu sağlamıştır.

Sistem Çözümleyici(Yüksek Başarılı): Bilgi toplama konusunda yüksek beceri göstermiş ve Regex ifadelerinin çözümlenmesini sağlamıştır.

Kalite Kontrol Uzmanı (Düşük Başarılı): Sistem müşteri ihtiyaçlarını tam olarak anlayamadığı için test konusunda yetersiz kalmıştır.

Yazılım Ekip Lideri(Yüksek Başarılı): Yazılım geliştirme sürecinde etkin rol oynamıştır.

Donanım Ekip Lideri(Yüksek Başarılı): Kullanılacak donanımları verimli bir şekilde tespit etmiş ve kullanmıştır.

Bu değerlendirmeler Melahat Erbaş tarafından tarafsız bir şekilde yapılmıştır.

- **Proje Yöneticisi**

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

- **Kalite Kontrol Uzmanı**

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

- **UI Tasarımcı**

Yapılan analizlere göre kullanıcıların deneyimlerinin iyileştirilmesini hedefler.

- **Donanım Ekip Lideri**

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

- **Yazılım Ekip Lideri**

Yazılım geliştirme sürecindeki baş aktördür. Bütün kullanıcı senaryolarını oluşturarak yazılım tasarımı ile birlikte kodlamayı geliştirir.

- **Sistem Çözümleyici**

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

- **Sistem Tasarımcı**

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

- **Sistem Yöneticisi**

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

- **Veri Tabanı Yöneticisi**

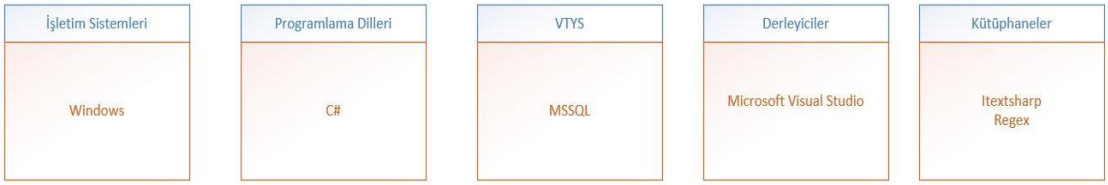
Veri tabanı sistemlerinin kurulması, konfigürasyonun yapılması, tasarlanması, sorgulanması ve güvenliğinin sağlanması işlemlerini üstlenmiştir.

2.5 Önerilen Sistemin Teknik Tanımları

Kullanılan Teknoloji

- Rx Kütüphanesi

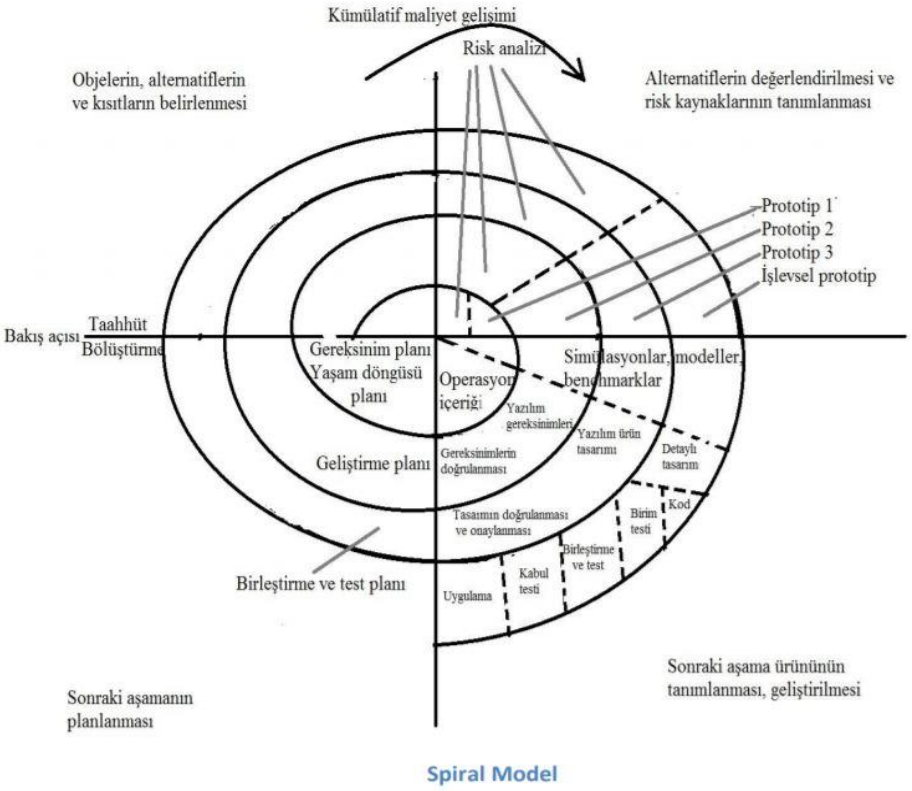
2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları



Şekil 2.6.1 Kullanılan araçlar

2.7 Proje Standartları Yöntem ve Metadolojiler

Spiralin başladığı ilk çeyrek içinde ilk isterler toplanır ve buna göre proje planlaması yapılır. İkinci çeyrekte, ilk tanımlanan isterlere göre risk çözümlemesi yapılır. Üçüncü çeyrekte, risk çözümlemesi sonunda ortaya çıkan isterlerin tanımlanmasındaki belirsizlikleri ortadan kaldırmak için prototipleme yöntemi kullanılır. Dördüncü çeyrekte, müşteri, ortaya çıkan ilk ürünü inceleyerek değerlendirme yapar, önerilerde bulunur. Bu şekilde tanımlanan ilk döngü bir sonraki döngü için bir girdi oluşturur.



Şekil 2.7.1 Proje geliştirme modeli

Aşama	Kullanılan Yöntem/Araçlar	Ne İçin Kullanıldığı	Çıktı
Planlama	- Veri Akış Şemaları, - Süreç Belirtimleri, - Görüşme, - Maliyet Kestirim Yöntemleri - Proje Yönetim Araçları	- Süreç İnceleme - Kaynak Kestirimi - Proje Yönetimi	Proje Planı
Çözümleme	- Süreç Belirtimleri, - Veri Akış Şemaları, - Görüşme, - Nesne İlişki Şemaları, - Veri Sözlüğü	- Süreç Çözümleme - Veri Çözümleme	Sistem Çözümleme Raporu
Çözümlemeden Tasarıma Geçiş	- Akışa Dayalı Çözümleme, - Süreç Belirtimlerinin Program Tasarım Diline Dönüştürülmesi - Nesne İlişki Şemalarının Veri Tablolarına Dönüştürülmesi	- Başlangıç Tasarım - Ayrıntılı Tasarım - Başlangıç Veri - Tasarımı	Başlangıç Tasarım Raporu
Tasarım	- Yapısal Şemalar - Program Tasarım Dili - Veritabanı Tabloları - Veri Sözlüğü	- Genel Tasarım - Ayrıntılı Tasarım - Veri Tasarımı	Sistem Tasarım Raporu

Şekil 2.7.2 Proje aşamaları

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

2.8 Kalite Sağlama Planı

Buna rağmen, yazılım kalitesi basit bir yolla tanımlanması mümkün olmayan karmaşık bir kavramdır. “Klasik olarak, kalite kavramı, üretilen ürünün belirtimlerini karşılaması gerektiğini ortaya koyar (Crosby, 1979).”

Kalite sağlama üzerine geliştirilmiş belirtimler gerçek dünyadaki çoğu ürün için uygulanabilse de yazılım için istenilen seviyeye ulaşamamış, tam olarak kaliteyi ölçmekte yararlı olamamışlardır. Bu projede yazılım kalite yönetimi üç temel davranış ile yapılandırılacaktır:

Kalite Sağlama Planı

1. **Kalite Güvence**: Yüksek kaliteye sahip yazılıma götürecek kurumsal yordam ve standartlar çatısının ortaya konması.
2. **Kalite Planlama**: Bu çatıdan uygun standart ve yordamların seçimi ve bunların projeye uyarlanması.
3. **Kalite Kontrol**: Proje kalite standart ve yordamlarının yazılım geliştirme takımı tarafından takip edildiğini garanti eden süreçlerin tanımlanması ve belgelenmesi.

2.8.1 Kalite Güvence

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

Kalite güvence sürecinin bir parçası olarak yerleştirilmiş iki tip standart vardır:

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

2. Süreç Standartları: Bunlar yazılım geliştirme süresince takip edilecek süreçleri tanımlayan standartlardır. Belirtim, tasarım ve doğrulama süreçlerinin tanımları ve bu süreçler süresince oluşturulması gereken belgelerin tanımları bunlara dahildir.

2.8.2 Kalite Planlama

Kalite planlama yazılım sürecinin erken bir evresinde başlanacaktır. Kalite planı istenen ürün kalitelerini ortaya koyar. Bunlara nasıl değer biçileceğini de tanımlamalıdır. Bu yüzden yüksek kaliteli yazılımın gerçekte ne anlama geldiğini tanımlar. Böyle bir tanım olmazsa, farklı mühendisler zıt yönlerde çalışabilirler böylece farklı ürün özellikleri en iyilenir. Kalite planlama sürecinin sonucu bir proje kalite planıdır.

Projede şu taslak üzerine gidilecektir:

1. Ürün başlangıcı: Ürünün, sunulacak pazarın ve ürün için kalite beklentilerinin tanımı

2. Ürün planları: Önemli sürüm tarihleri ve ürün sorumlulukları yanında dağıtım ve ürün bakım planları

3. Süreç tanımlamaları: Ürün geliştirme ve yönetimi için kullanılacak geliştirme ve bakım süreçleri

4. Kalite hedefleri: Önemli ürün kalite özelliklerinin de tanımlandığı ürün kalite hedef ve planları

5. Riskler ve risk yönetimi: Ürün kalitesini etkileyebilecek anahtar riskler ve bu riskleri karşılayan eylemler. Kalite planları yazılırken bunların olabildiğince kısa olmasına dikkat edilecektir.

2.8.3 Kalite Kontrol

Kalite kontrol, kalite güvence yordam ve standartlarına uyulmasını garanti etmek için yazılım geliştirme sürecinin gözden geçirilmesini gerektirir.

Kalite kontrol sürecinin yazılım geliştirme sırasında kullanılması gereken kendi yordam ve rapor kümesi vardır. Bu yordamlar yazılımı geliştiren mühendisler tarafından düzgün ve kolaylıkla anlaşılabilir olmalıdır.

Kalite kontrole iki çeşit tamamlayıcı yaklaşım vardır:

- Yazılımı geliştirmek için kullanılan belgeleme ve süreçlerin kalite incelemesi bir grup insan tarafından yapılacaktır. Bu kişiler proje standartlarının izlenmesinin ve yazılım ile belgelerin standartlara uygunluğunun kontrolünden sorumlu olacaktır. Standartlardan sapmalar kayıtlanmalı ve proje yönetiminin dikkatine sunulmalıdır.
- Üretilen yazılım ve standartların bir program tarafından yapılır ve geliştirme projesine uygulanan standartlarla karşılaştırılırsa buna otomatikleştirilmiş yazılım değerlendirme adı verilecektir. Otomatikleştirilmiş değerlendirme bazı yazılım özelliklerinin nicel ölçümlerini gerektirir.

2.9 Konfigürasyon Yönetim Planı

Sistemin ilerde kullanıcının yeni istemlerini karşılayamaması veya sistemin yapısındaki bazı bileşenlerin değişmesi sonucu güncelliğini kaybettiğinde olası konfigürasyon planı hazırlandı.

- **Veri girişinin hatalı yapılması**: Kullanıcının yüklediği dosya türünün belirlenen dosya türünden farklı olması halinde gerçekleşir.
- **Kullanılan tanımlamaların dışına çıkılması**: Tanımlanan rx ifadesinin dokümanı analiz edememe durumu veya yanlış analiz etmesidir.

Durumları için konfigürasyon yönetim planı oluşturuldu.

2.10 Kaynak Yönetim Planı

Mevcut bir kaynağımız olmadığından kaynak olarak sadece bu proje dokümantasyonu var. Bunun yanında tez örneğinden yola çıkarak mevcut sistemi geliştirdik [1].

2.11 Eğitim Planı

Projeden kazanılacak en önemli olaylardan biride eğitimdir. Kullanılacak dillerin arayüz editör ve programların kullanımında hakim olunamaması halinde bu program başarıyla neticelendirilemez. Bu yüzden projede bazı eğitimler alınması gereklidir. Proje kapsamında alınacak olan eğitimler;

- C# Dil Eğitimi
- Rx Kütüphanesi Eğitimi
- Microsoft Visual Studio Kullanım Eğitimi
- SQL Dil Eğitimi

Gereken eğitimlerdir.

2.12 Test Planı

Proje test ekipleri ve görevleri şu şekildedir;

- Sistem tasarımcıları, kullanıcı olarak da görev yapabilecektir.
- Ekibin test aşamasında yapacağı işler aşağıdaki gibidir;
- Sistem tasarımcıları program için uygun veri setlerinin girişini yaparak, sistem içerisindeki her bir fonksiyonun çalışma durumunu analiz edeceklerdir. Pilot belgede bir uygulama gerçekleştireceklerdir.

2.13 Bakım Planı

Sistem ilk 1 sene 3 ayda bir bakıma alınacaktır. Kullanıcıların geri dönütleri sayesinde sistem iyileştirilmeye çalışılacaktır.

2.14 Projede Kullanılan Yazılım/Donanım Araçlar

"2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları" başlığında projede kullanılan yazılım donanım araçları belirtilmiştir.

3. Sistem Çözümleme

3.1 Mevcut Sistemin İncelenmesi

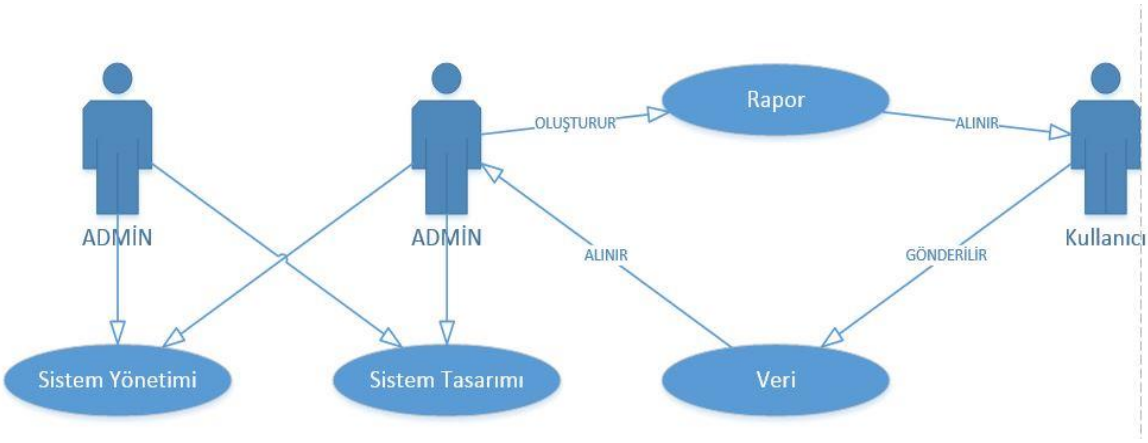
Projenin gereksinimlerin araştırılması, tanımlanması, ortaya çıkarılması ve açıklanması yapılacaktır.

Örnek dokümanın gerek syntax yapısı gerekse semantiği incelenerek yapılması hedeflenen sisteme nasıl entegre edileceği, işleneceği araştırılmış ve sisteme uygulanmıştır [1].

3.1.1 Örgüt Yapısı

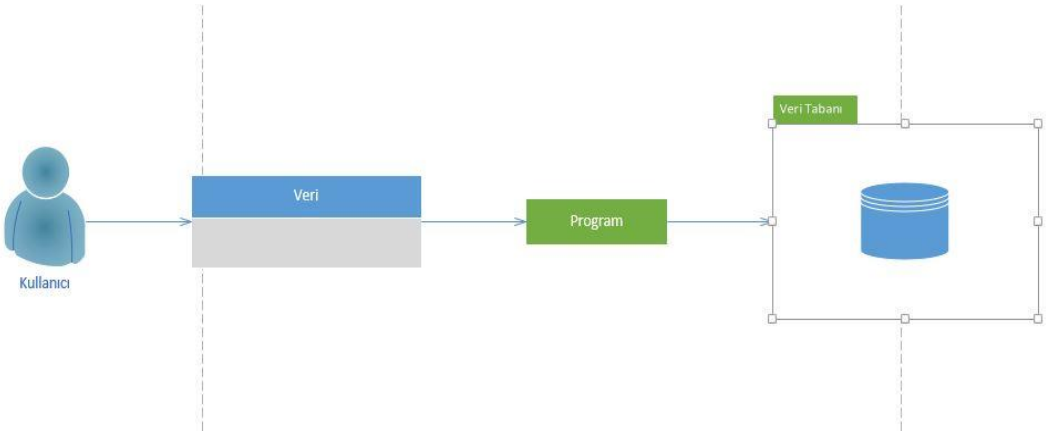
Bir örgüt yapısı bulunmamaktadır.

3.1.2 İşlevsel Model



Şekil 3.1.2.1 Sistem Use-case Diyagramı

3.1.3 Veri Modeli

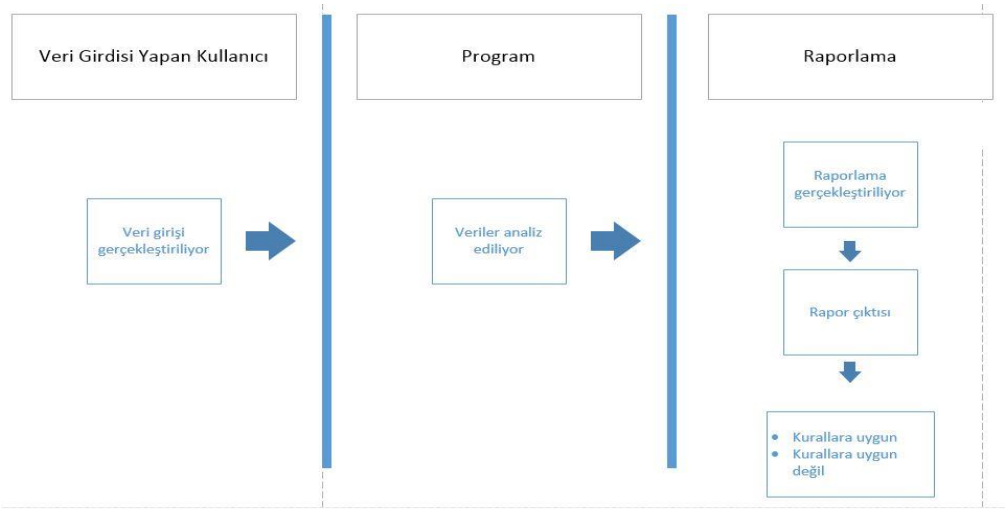


Şekil 3.1.3.1 Sistem veri modeli

3.1.4 Varolan Yazılım/Donanım Kaynakları

- Microsoft Visio
- Ms Word
- Pdf
- Regex

Şekil 3.1.4.1 Donanımsal kaynaklar



Şekil 3.1.4.2 Veri transferi ve raporlama

3.1.5 Varolan Sistemin Değerlendirilmesi

Programa girişi yapılacak olan yanlış dosya tipinin engellenmesi için filtreleme kullanılacaktır.

```
public string veriOku()
{
    openFileDialog1.ShowDialog();
    openFileDialog1.Filter = "Pdf Dosyası |*.pdf";
    return openFileDialog1.FileName;
}
```

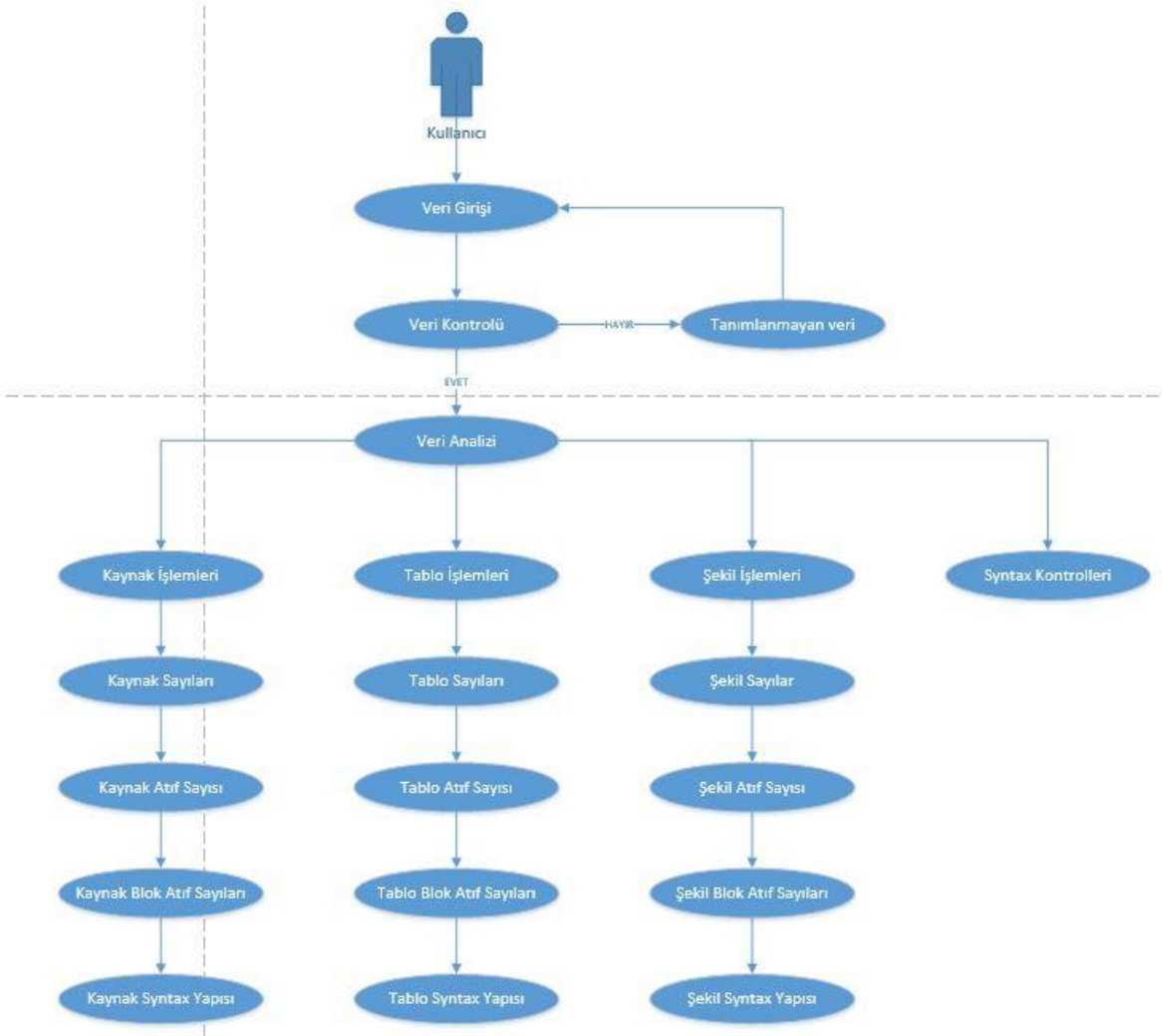
Şekil 3.1.5.1 Filtreleme kod parçacığı

3.2 Gereksenen Sistemin Mantıksal Modeli

3.2.1 Giriş

Bu bölümde önerilen sistemin işlevsel yapısı, veri yapısı ve kullanıcı ara yüzünde çözümleme yapılır. Bu model baz alınarak sistem tasarlanır.

3.2.2 İşlevsel Model



Şekil 3.2.2.1 Use-Case diyagramı 1

Şekil 3.2.2.2 Use-Case Diyagramı 2

Senaryo Adı: Veri Girişi
Aktör: Kullanıcı
Amaç: Veri Tip Kontrolleri
Ana Senaryo:

- Veri girişi yapılır
- Veri kontrolü yapılır
- Eğer veri tipi uygun ise veri analizine geçilir.
- Değil ise tekrar veri girişine gönderilir

Senaryo Adı: Kaynak İşlemleri
Aktör: Kullanıcı
Amaç: Kaynak Kontrolleri
Ana Senaryo:

- Kaynak işlemleri sekmesine girilir
- Kaynak sayıları hesaplanır
- Kaynak atıf sayısı hesaplanır
- Kaynak blok atıf sayıları hesaplanır
- Kaynak syntax yapısı kontrol edilir
- Raporlama yapılır

Senaryo Adı: Tablo İşlemleri
Aktör: Kullanıcı
Amaç: Tablo Kontrolleri
Ana Senaryo:

- Tablo işlemleri sekmesine girilir
- Tablo sayıları hesaplanır
- Tablo atıf sayısı hesaplanır
- Tablo blok atıf sayıları hesaplanır
- Tablo syntax yapısı kontrol edilir
- Raporlama yapılır

Senaryo Adı: Şekil İşlemleri
Aktör: Kullanıcı
Amaç: Şekil Kontrolleri
Ana Senaryo:

- Şekil işlemleri sekmesine girilir
- Şekil sayıları hesaplanır
- Şekil atıf sayısı hesaplanır
- Şekil blok atıf sayıları hesaplanır
- Şekil syntax yapısı kontrol edilir
- Raporlama yapılır

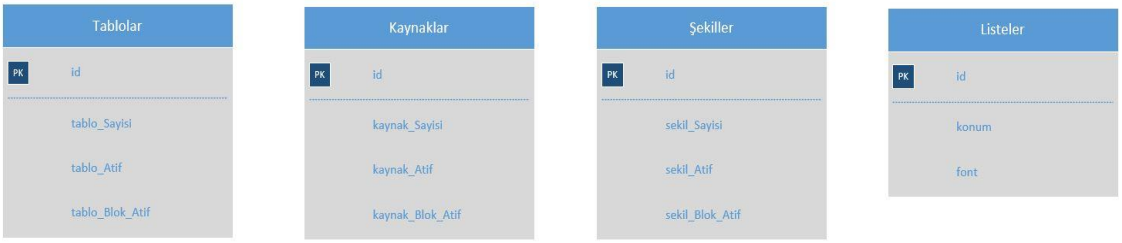
Senaryo Adı: Syntax Kontrolleri
Aktör: Kullanıcı
Amaç: Şekil Kontrolleri
Ana Senaryo:

- Syntax Kontrolleri sekmesine girilir
- Ad Soyad yazımı kontrol edilir
- Tez başlığı yazımı kontrol edilir
- Bilim dalı kontrol edilir
- Tarih yazımı kontrol edilir
- Danışman ismi kontrol edilir
- Danışman kurumu yazımı kontrol edilir
- 2. Danışman var mı kontrol edilir
- 2. Danışman kurumu yazımı kontrol edilir
- Tez ilk teslimde tez onay sayfası varlığı kontrol edilir
- Önsöz de teşekkür konumu kontrol edilir
- Paragraf girintisi kontrol edilir
- İçindekiler kısmında ana başlık yazımı kontrol edilir
- Alt başlıklar girintisi kontrol edilir(İçindekiler kısmı)
- İçindekiler sayfa sayısı konumu kontrol edilir
- Şekiller listesi etiketler konumu kontrol edilir
- Şekiller listesi sayfa numarası konumu kontrol edilir
- Tablolar listesi etiketler konumu kontrol edilir
- Tablolar listesi sayfa numarası konumu kontrol edilir
- Şekiller ve tablolar listesi etiketler yazımı kontrol edilir
- Ekler sayfası varlığı kontrol edilir
- Ekler listesi varlığı kontrol edilir
- Ekler listesi etiketler konumu kontrol edilir
- Ekler listesi sayfa numarası konumu kontrol edilir
- Simgeler ve kısaltmalar yazımı kontrol edilir
- Ana başlık sayfa numarası varlığı kontrol edilir
- Tablo ve şekiller etiket ve başlık yazımı kontrol edilir
- Denklem numaraları ile başlık uyumu kontrol edilir
- Denklemlere yapılan atıf kontrol edilir
- Tablolar için etiket ve başlık konumu kontrol edilir
- Şekiller için etiket ve başlık konumu kontrol edilir
- Türkçe kaynaklar için sayfa sayısı kısaltması kontrol edilir
- Yabancı kaynaklar için sayfa sayısı kısaltması kontrol edilir
- Özgeçmişte fotoğraf varlığı kontrol edilir
- Özgeçmişte iş deneyimi varlığı kontrol edilir
- E-posta yazımı kontrol edilir
- Çift tırnak arasındaki kelime sayısını elliden fazla olup olmadığı kontrol edilir
- Paragraf satır sayılarını ikiden fazla olup olmadığı kontrol edilir
- Kaynaklarda dergiler için yazımın doğru olup olmadığı kontrol edilir
- Kaynaklarda kitaplar için yazımın doğru olup olmadığı kontrol edilir
- Kaynaklarda web siteler için yazımın doğru olup olmadığı kontrol edilir
- Raporlama yapılır

3.2.3 Genel Bakış
3.2.4 Bilgi Sistemleri/Nesneler

Kullanıcı: Veri girişi yapar. Analizler sonucu geri dönüt olarak aldığı rapordan yapmış olduğu hataları tespit edebilir.
Veri: Kullanıcı programa giriş olarak verir. Belirlenen tip pdf olmasına karşın hatalı tip girişinde kullanıcıdan yeni veri girişi istenir.

3.2.5 Veri Modeli



Şekil 3.2.5.1 Veri modeli

3.2.6 Veri Sözlüğü

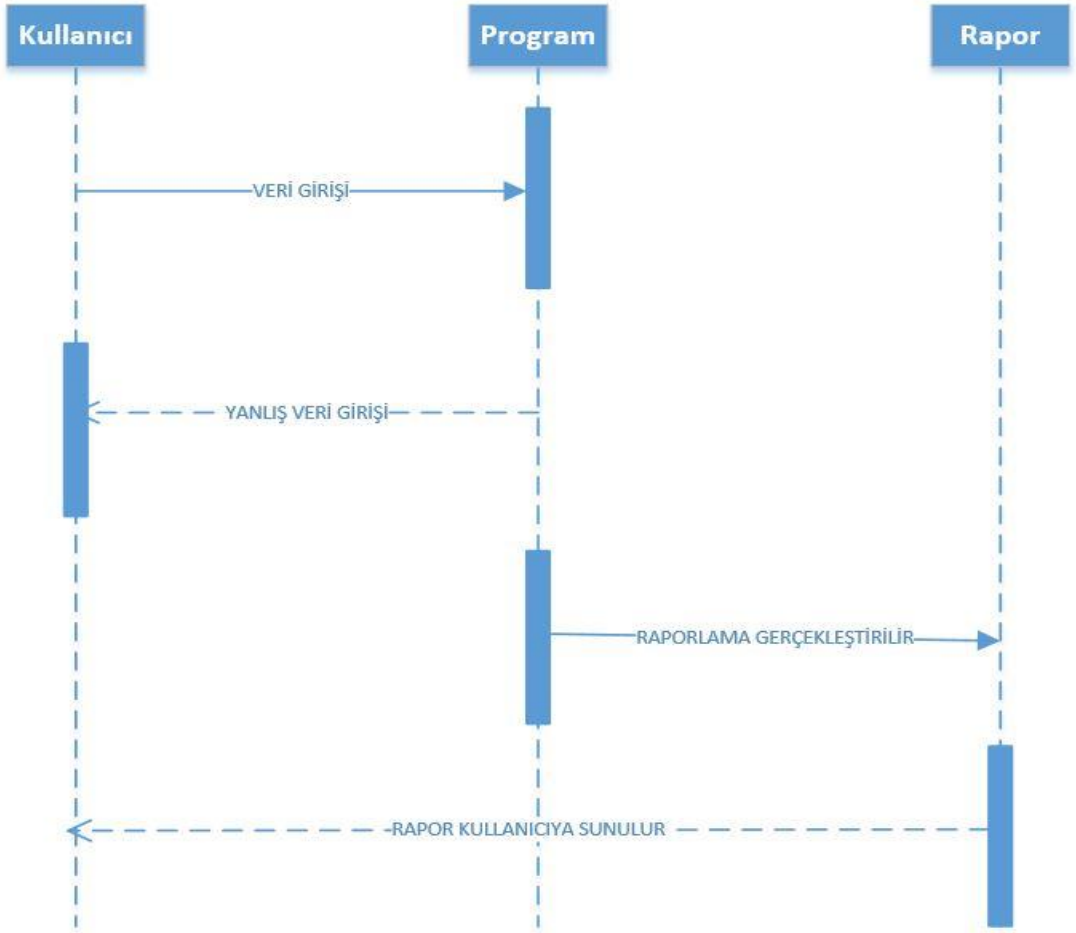
Tablolar
- id: İşlenen veride kayıt sıralaması
- tablo_Sayısı: İşlenen verideki tablo sayısını tutar.
- tablo_Atif: İşlenen verideki tablo atıf sayısını tutar.
- tablo_Blok_Atif: İşlenen veride tablo blok atıf sayısını tutar

Kaynaklar
- id: İşlenen veride kayıt sıralaması
- kaynak_Sayısı: İşlenen verideki kaynak sayısını tutar
- kaynak_Atif: İşlenen verideki kaynak atıf sayısını tutar
- kaynak_Blok_Atif: İşlenen verideki kaynak blok atıf sayısını tutar

Şekiller
- id: İşlenen veride kayıt sıralaması
- sekil_Sayısı: İşlenen verideki şekil sayısını tutar
- sekil_Atif: İşlenen verideki şekil atıf sayısını tutar
- sekil_Blok_Atif: İşlenen verideki şekil blok atıf sayısını tutar

Listeler
- id: İşlenen veride kayıt sıralaması
- konum: Liste verilerini tutar

3.2.7 İşlevlerin Sıra Düzeni



Şekil 3.2.7.1 İşleyiş Diyagramı

Şekil 3.2.7.2 Etkinlik Diyagramı

3.1.8 Başarım Gerekleri

Mevcut sistemler incelendi ve mevcut sistemin eksikliklerinden yola çıkarak, sistemin başarımı için

- Tüm dosya türlerinin kabul edilmesi
- Raporlama kısmına ek olarak doküman içerisindeki hataların düzeltilmesi
- Kullanıcı kayıt formu oluşturulması
- Hataları düzeltilmiş dokümanın indirilebilir hale getirilmesi

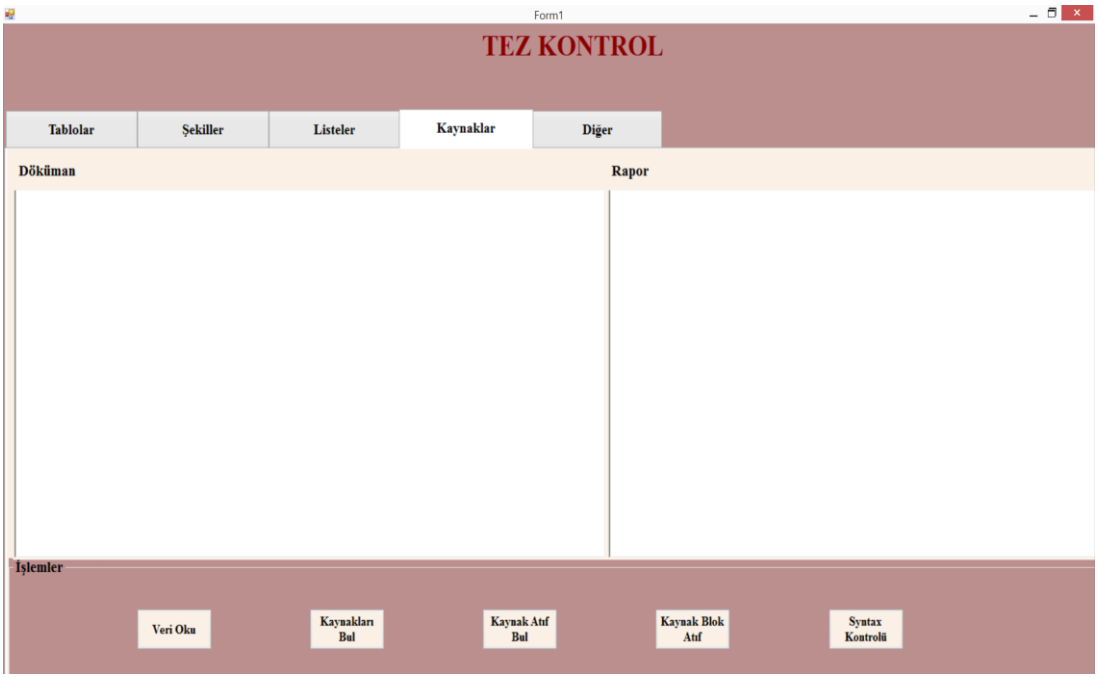
Temel gereklilikler olarak tespit edilmiştir.

3.2 Arayüz (Modül) Gerekleri

3.2.8 Yazılım Arayüzü

Projenin çalışması esnasında yazılım arayüzünde açık oluşmamasına özen gösterildi. Gerekli olan her türlü değişiklik kaynak kodları üzerinden yapılıp tekrar derlenecek.

3.2.9 Kullanıcı Arayüzü



Şekil 3.2.9.1 Kaynaklar Modülü Kullanıcı Arayüzü

3.2.10 İletişim Arayüzü

Geri dönüşler kullanıcılardan e-mail ile alınacaktır.

3.2.11 Yönetim Arayüzü

Projede herhangi bir yönetim arayüzü bulunmamaktadır.

3.3 Belgeleme Gerekleri

3.3.8 Geliştirme Sürecinin Belgelenmesi

Geliştirme sürecinde yapılan belgelendirmenin amacı projenin tamamlanma yüzdesini anlamak ve projenin müşteri istekleri ile olan uygunluğunu ölçmek için müşteriye sunulmasıdır.

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

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

3.3.9 Eğitim Belgeleri

Mevcut bir eğitim belgesi bulunmamaktadır.

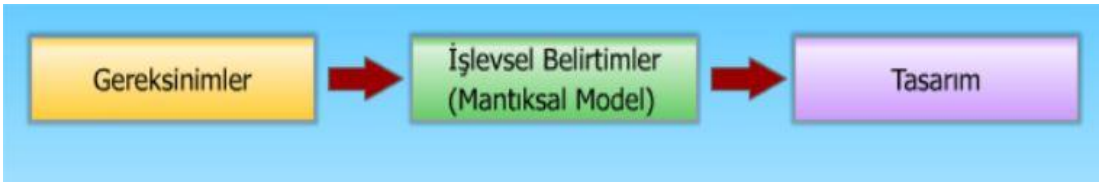
3.3.10 Kullanıcı El Kitapları

Kullanıcı el kitapları bulunmamaktadır.

4 Sistem Tasarımı

4.1 Genel Tasarım Bilgileri

4.1.1 Genel Sistem Tanımı



Şekil 4.1.1.1 Genel sistem tanımı

- **Gereksinimler**

Örnek olarak aldığımız tez dokümanı içerisinden müşteri ve proje yöneticileri işbirliği ile gereksinimler çıkarılmıştır. Çıkarılan gereksinimlerin işlevsel modele aktarılması ile tasarım modülü oluşturulmuştur.

- **İşlevsel Belirtimler**

Sistem ne yapacak?

Kullanıcıdan veri pdf türünde veri girişi alacak ve çıktı olarak pdfte bulunan hataları gösterecektir.

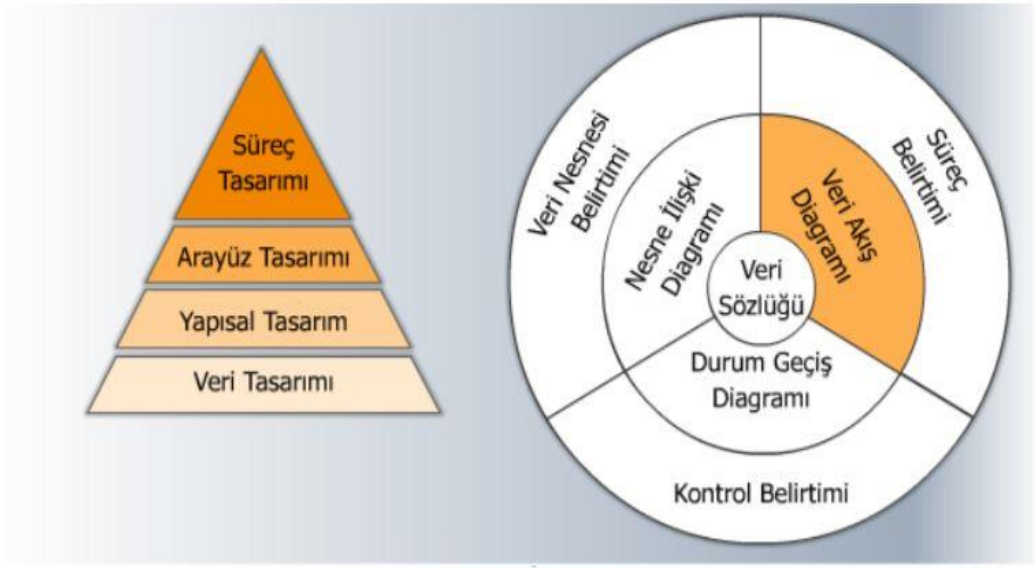
Yazılım ne yapacak?

Kullanıcıdan aldığı pdf türündeki verinin belirlenen syntax ve semantic kurallarına uygunluğunu kontrol edecektir.

- **Tasarım**

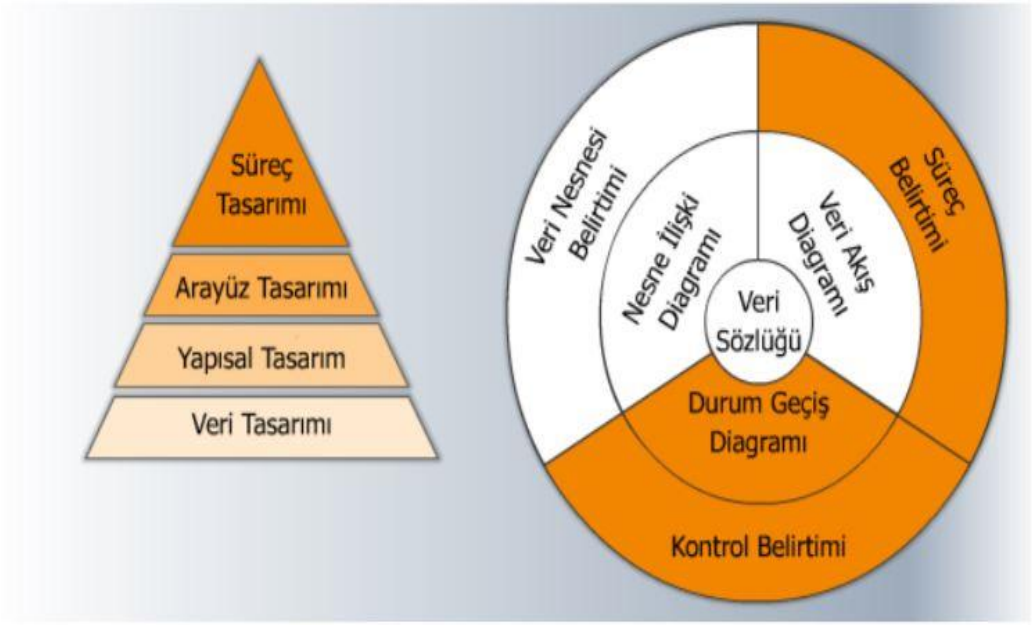
Tasarım aşamaları aşağıdaki gibi ilerleyecektir.

Süreç Tasarımı Aşamaları



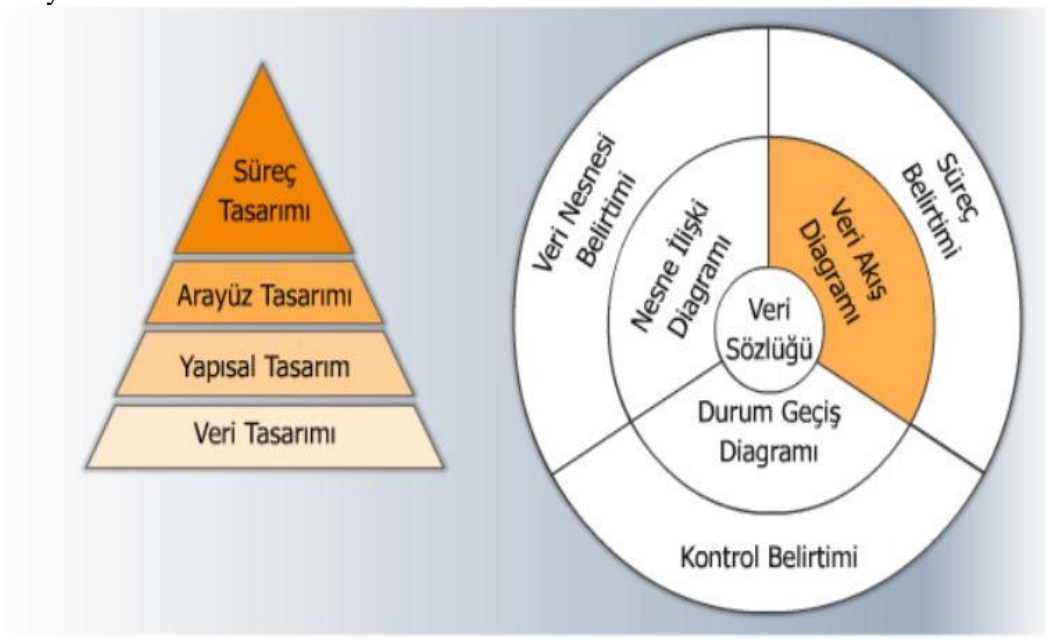
Şekil 4.1.1.2 Tasarım Aşamaları 1

Süreç Tasarımı



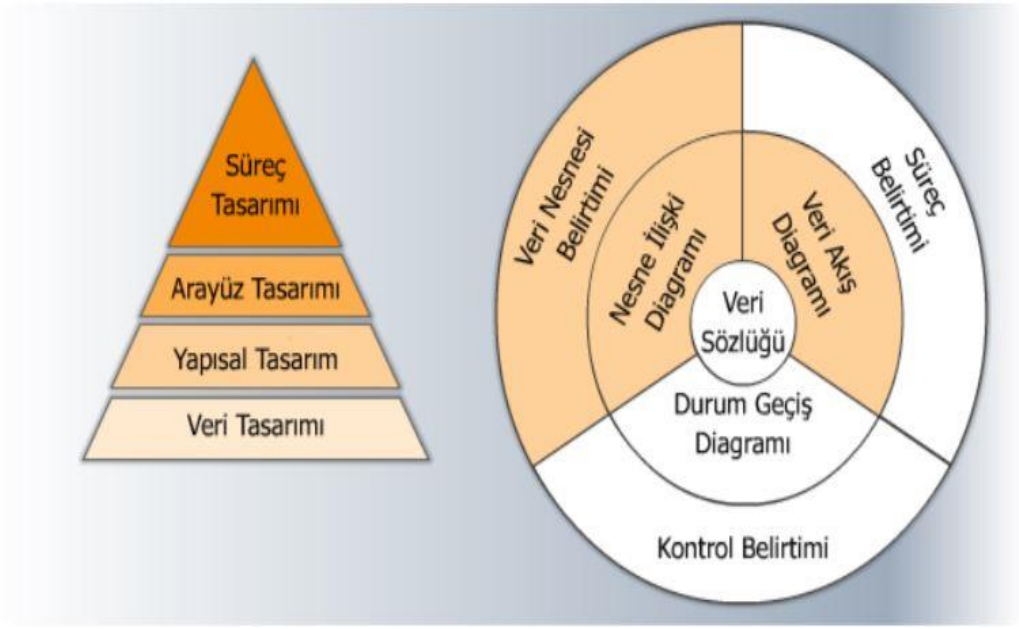
Şekil 4.1.1.3 Tasarım Aşamaları 2

Arayüz Tasarımı



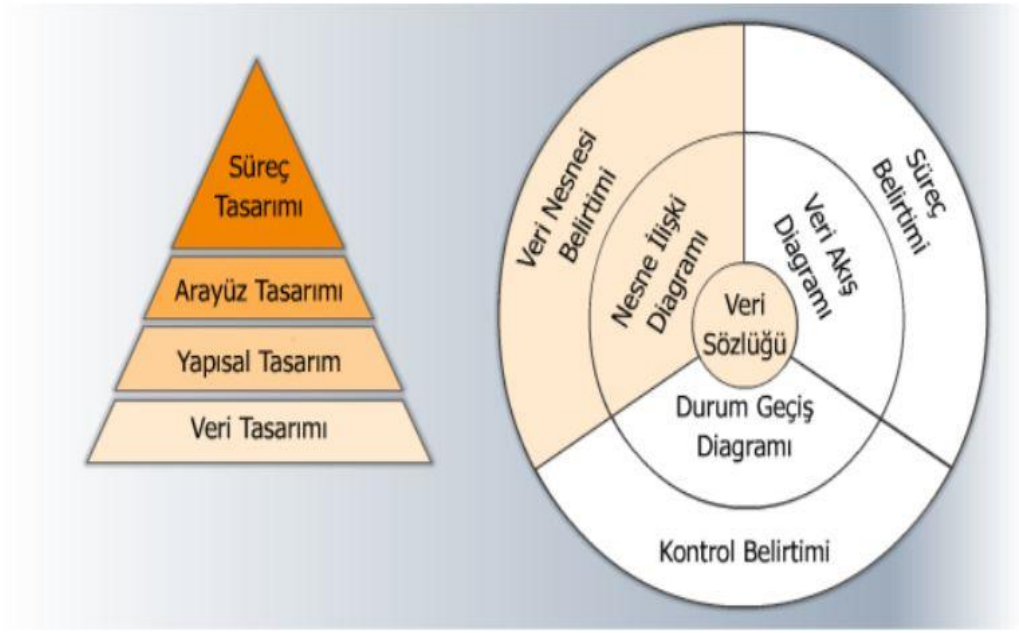
Şekil 4.1.1.4 Tasarım Aşamaları 3

Yapısal Tasarım



Şekil 4.1.1.5 Tasarım Aşamaları 4

Veri Tasarımı



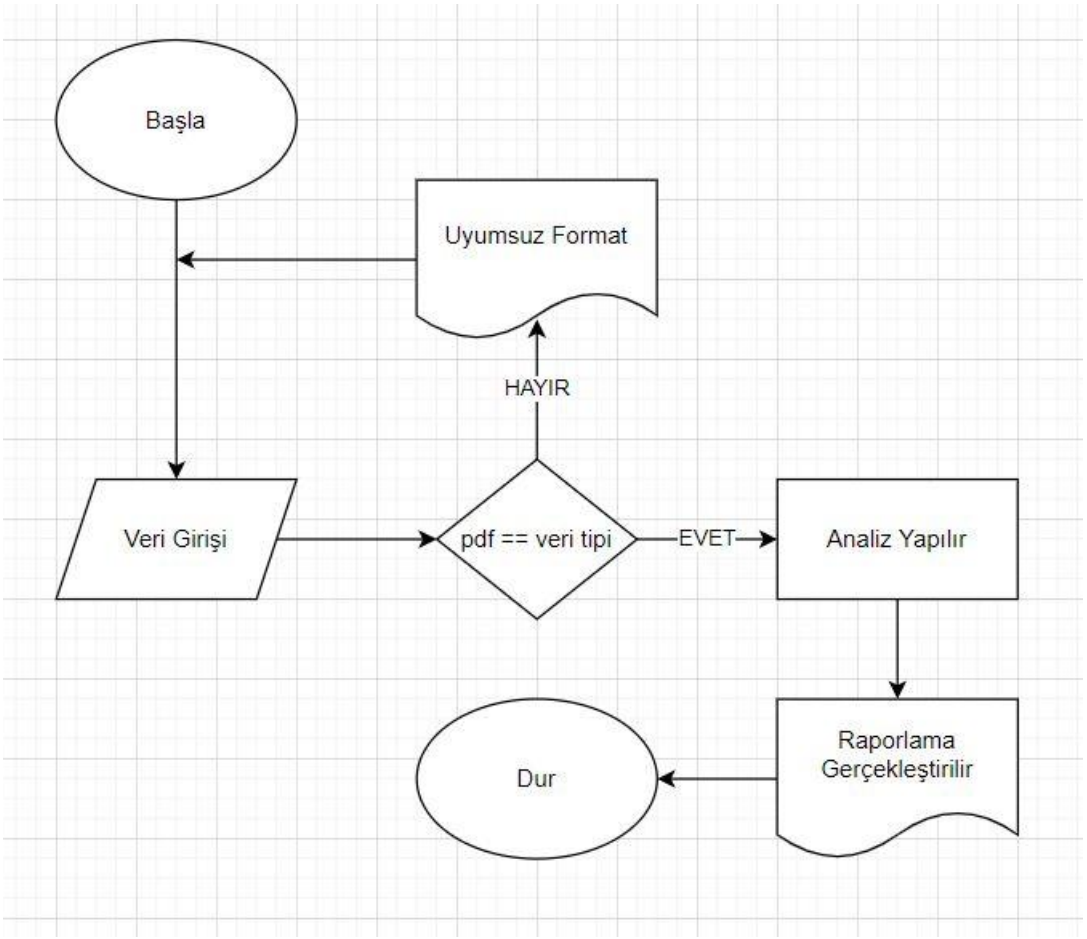
Şekil 4.1.1.6 Tasarım Aşamaları 5

4.1.2 Varsayımlar ve Kısıtlamalar

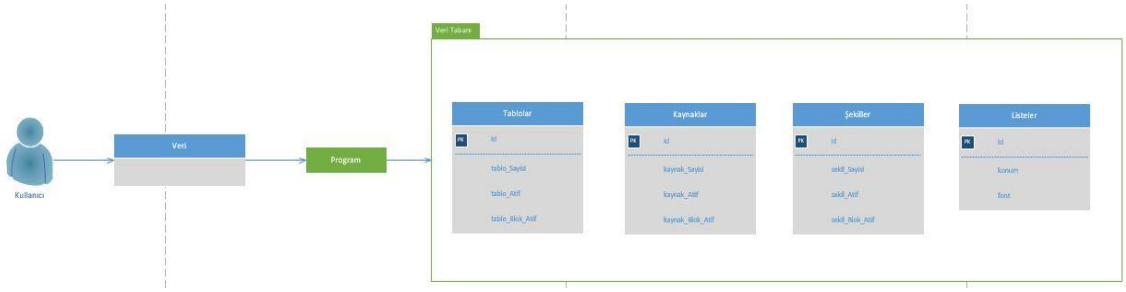
Sistemde varsayımlar bulunmamaktadır. Kısıtlar ise şu şekildedir:

- Veri girişi yapılan dosya türünün pdf olması gerekmektedir.
- Bütün raporlama işlemleri başlıklara ayrıldığı için toplu bir işlem yapılamaz.

4.1.3 Sistem Mimarisi

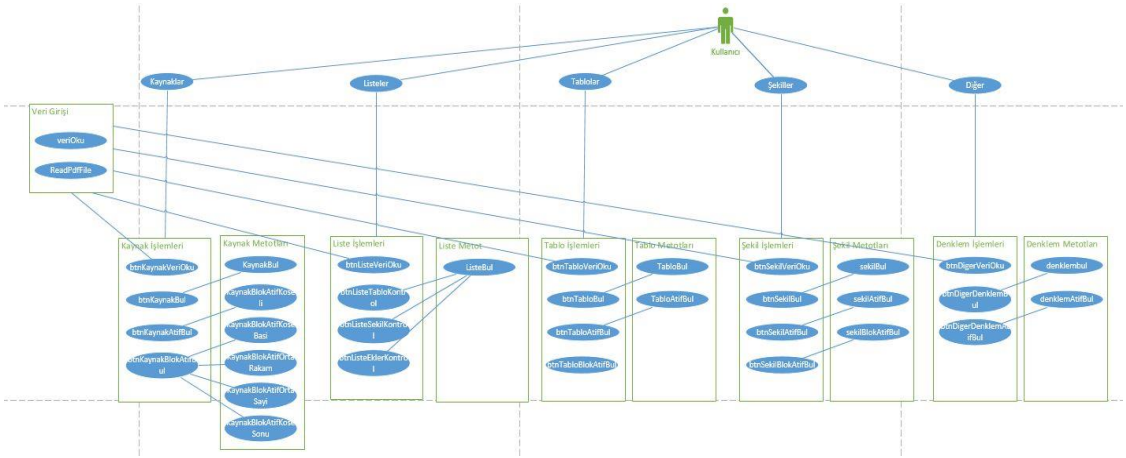


Şekil 4.1.3.1 Sistem mimarisi akış şeması



Şekil 4.1.3.2 Veri tabanı sistemi

Sistemin mimarisinin akış diyagramı şeklinde verilmesinin temel nedeni sistemin işleyiş mantığının nasıl olduğu ve nasıl bir yol çizileceğinin bilinmesidir.



Şekil 4.1.3.3 Sistem Mimarisi Uml Use Case

4.1.4 Dış Arabirimler

4.1.4.1 Kullanıcı Arabirimleri

Sisteme girişi için herhangi bir aktivasyon bulunmadığından dolayı, doğrudan sisteme erişim sağlanacaktır. Ayrıca her birimin kendine ait bir ekranı olacaktır.

4.1.4.2 Veri Arabirimleri

Veri arabirimlerinde SQL dili kullanılacaktır. Analiz edilen verilerin raporları veri tabanına kayıt edilecektir.

4.1.4.3 Diğer Sistemler Arabirimler

Başka bir arabirim kullanılmayacaktır.

4.1.5 Veri Modeli

Veri tabanındaki tablolar arası herhangi bir veri ilişkisi bulunmamaktadır.

4.1.6 Testler

Genel hatlarıyla testlerimiz iki aşamada gerçekleştirilecektir.

Alfa aşaması: Sistem geliştiricileri tarafından yapılan testleri içeren aşamadır.

Beta aşaması: Sistem sunulduktan sonra, kullanıcı geri dönütlerinin alındığı test aşamasıdır.

4.1.7 Performans

Sistemin performansını etkileyen faktörlerin test verileri değerlendirilecektir.

Sistemin Tasarıma Uygunluk Performansı;
Tasarımı yapılan sistemin stabilitesi ve işleyiş performansı değerlendirilecektir.

Veri Yapısının Sistemle Performansı;
Veri yapısının sistemle stabilitesi ve çalışma zamanındaki uyumluluk düzeyindeki performansı değerlendirilecek.

4.2 Veri Tasarımı

4.2.1 Tablo Tanımları

Sistem 4 tablodan oluşmaktadır. Bunun yanı sıra kullanıcının ekstra istekleri doğrultusunda ek tablolara açık bir sistem tasarlanmaktadır. “Tablolar”, ”Kaynaklar”, ”Şekiller”, ”Listeler” isimli tablolar oluşturulmuştur. Bu tablolar arasında herhangi bir ilişki bulunmamaktadır.

Tablo 4.2.1.1 Veri tabanı tablosu veri tipleri

Tablolar	Veri Tipi
id	İnt (Primary key) Identity
tablo_Sayisi	Varchar(50)
tablo_Atif	Varchar(50)
tablo_Blok_Atif	Varchar(50)

Kaynaklar	Veri Tipi
id	İnt (Primary key) Identity
kaynak_Sayisi	Varchar(50)
kaynak_Atif	Varchar(50)
kaynak_Blok_Atif	Varchar(50)

Tablo 4.2.1.2 Veri tabanı tablosu veri tipleri

Şekiller	Veri Tipi
id	İnt (Primary key) Identity
sekil_Sayisi	Varchar(50)
sekil_Atif	Varchar(50)
sekil_Blok_Atif	Varchar(50)

Listeler	Veri Tipi
id	İnt (Primary key) Identity
konum	Varchar(50)
font	Varchar(50)

4.2.2 Tablo İlişki Şemaları

Kullanılan veri tabanındaki tablolar arasında herhangi bir ilişki bulunmamaktadır.

4.2.3 Veri Tanımları

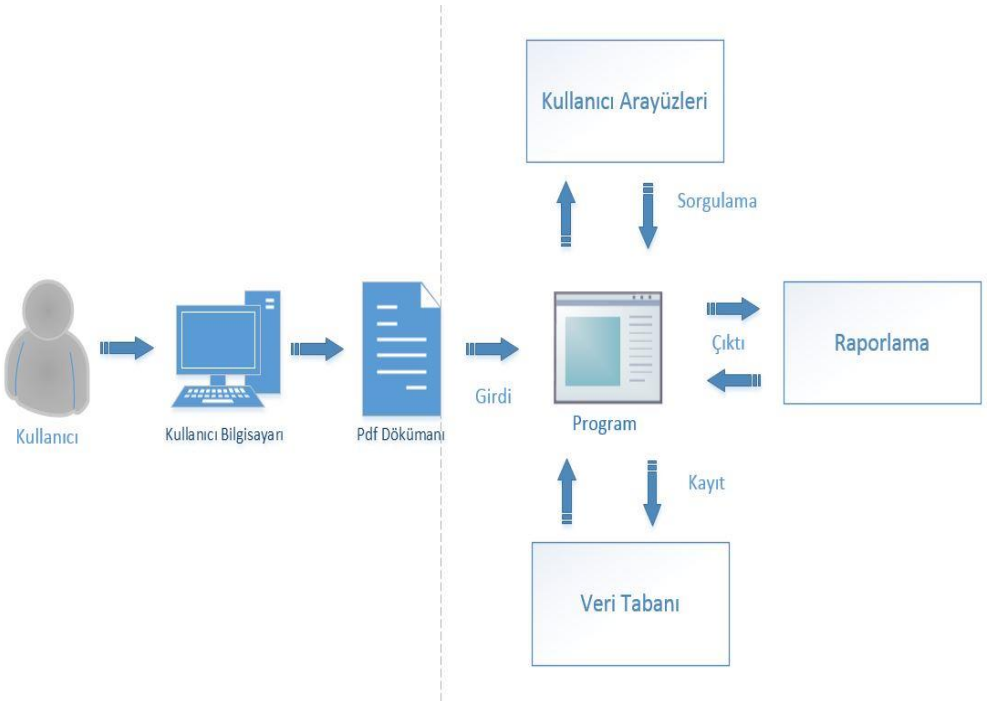
Veri tabanında id ‘lerin int olarak tanımlanması identity gerekliliğinden dolayıdır. Varchar(50) ile tanımlanan veriler ise matematiksel bir işlem içinde kullanılmayacağından ötürü bu şekilde tercih edilmiştir.

4.2.4 Değer Kümesi Tanımları

Değer kümesi tanımları belirtilmiştir. Değerleri girilen veri setine göre farklılıklar gösterebilir.

4.3 Süreç Tasarımı

4.3.1 Genel Tasarım



Şekil 4.3.1.1 Genel tasarım

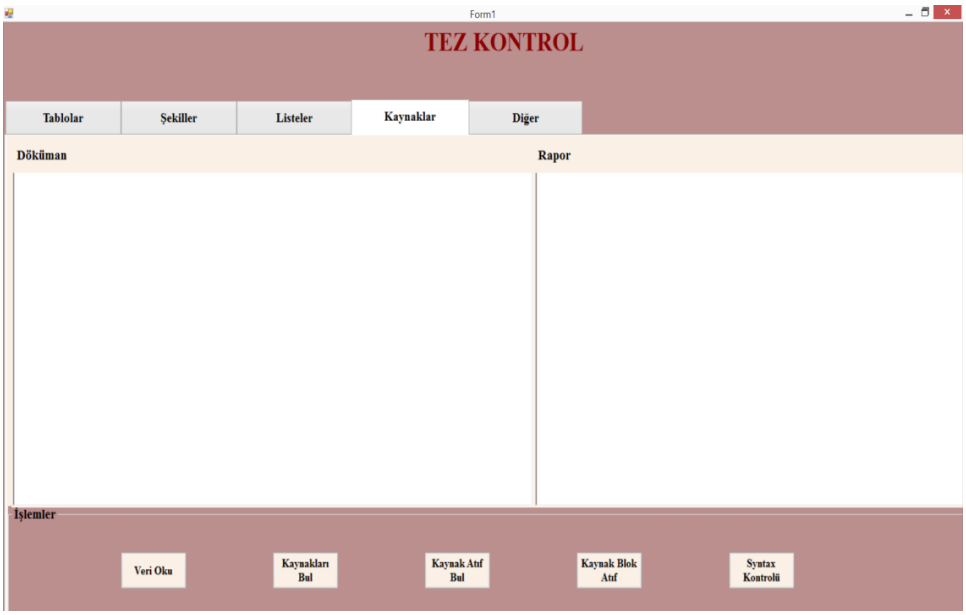
4.3.2 Modüller

4.3.2.1 Kaynaklar Modülü

İşlev

Kullanıcı tarafından yapılan girdinin kaynak kontrol işlemlerini gerçekleştirir.

Kullanıcı Arabirimi

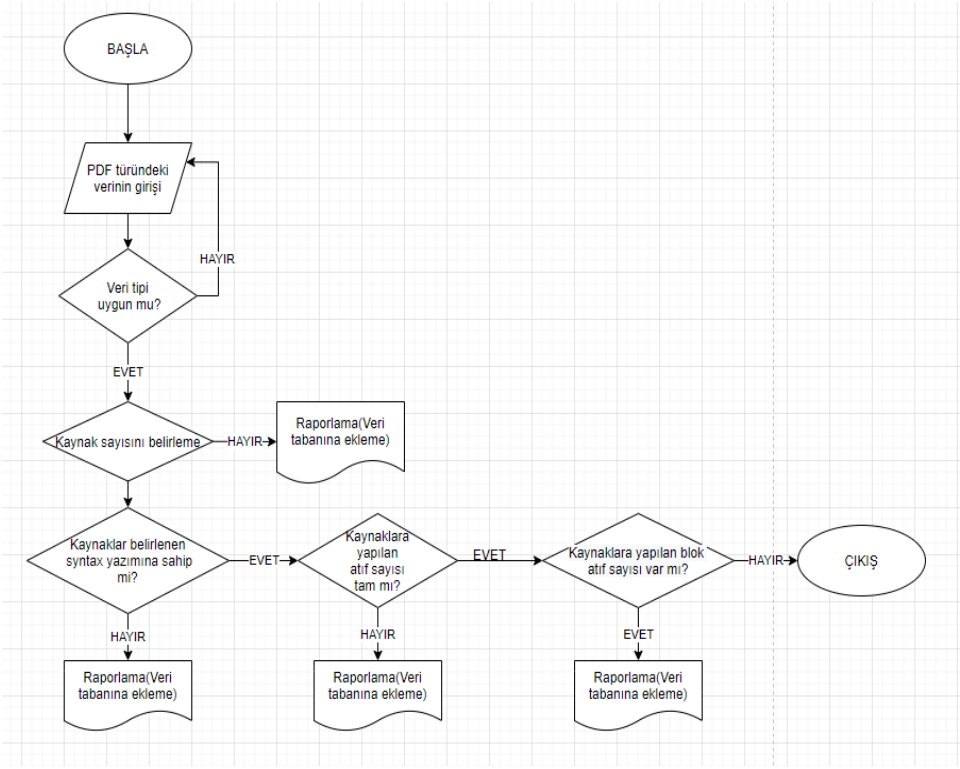


Şekil 4.3.2.1.1 Kaynaklar Modülü Kullanıcı Arabirimi

Modül Tanımı

Bu modül girdinin kaynak sayısını, kaynağın syntax yapısını, blok atıf sayısını ve atıf sayısını kontrol eder.

Modül İç Tasarımı



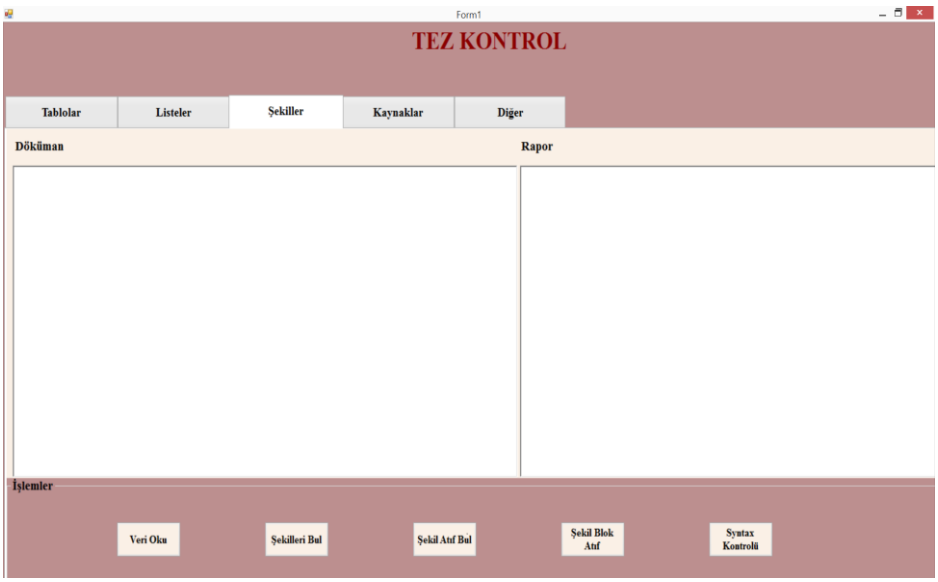
Şekil 4.3.1.4 Kaynaklar modülü akış diyagramı

4.3.2.2 Şekiller Modülü

İşlev

Kullanıcı tarafından yapılan girdinin şekil kontrol işlemlerini gerçekleştirir.

Kullanıcı Arabirimleri

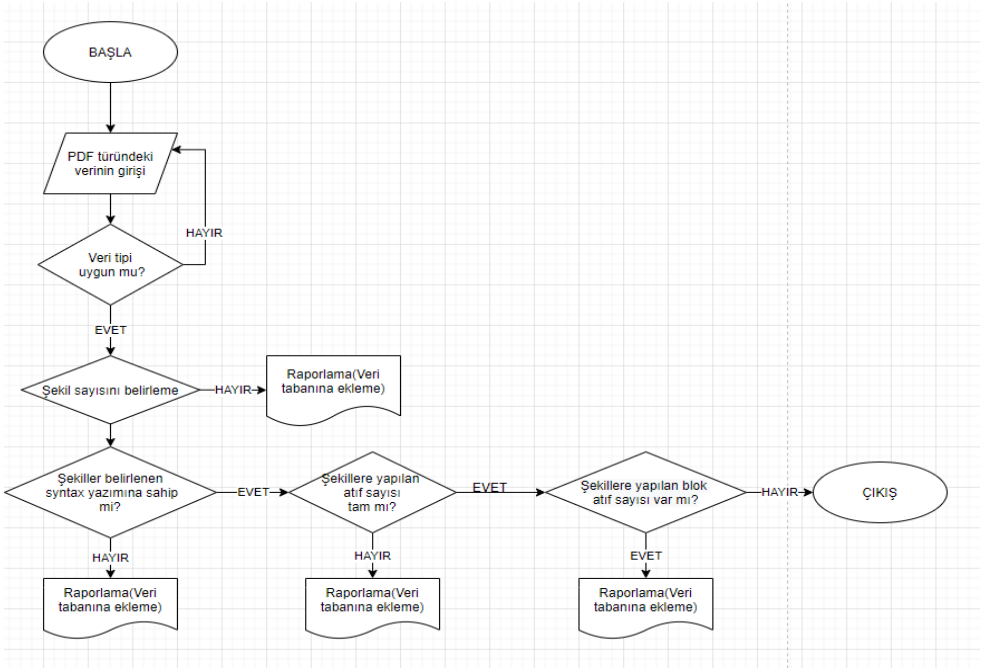


Şekil 4.3.2.2.1 Şekiller Modülü Kullanıcı Arabirimi

Modül Tanımı

Bu modül girdinin şekiller sayısını, şekiller syntax yapısını, blok atıf sayısını ve atıf sayısını kontrol eder.

Modül İç Tasarımı



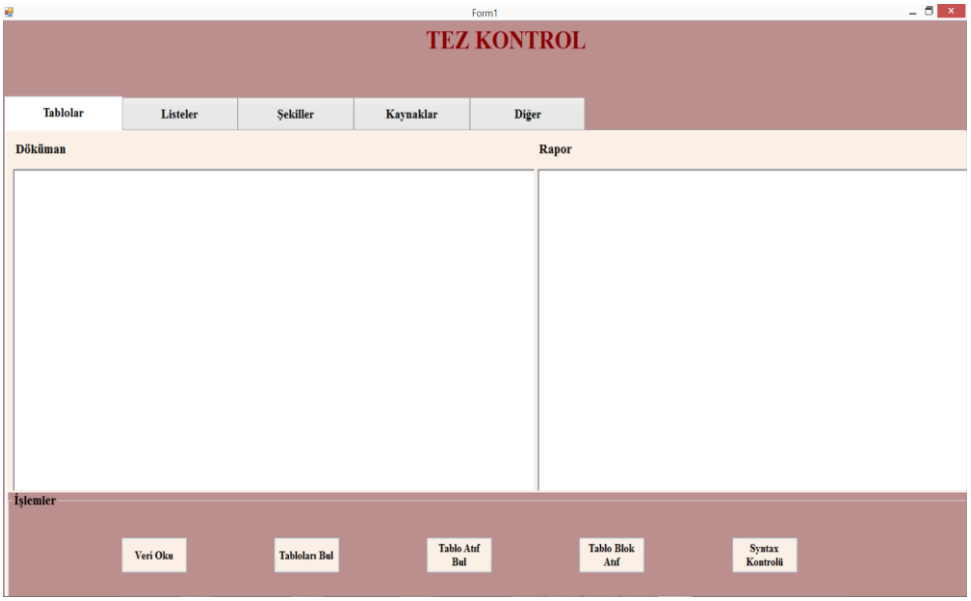
Şekil 4.3.1.2 Şekiller modülü akış diyagramı

4.3.2.3 Tablolar Modülü

İşlev

Kullanıcı tarafından yapılan girdinin tablo kontrol işlemlerini gerçekleştirir.

Kullanıcı Arabirimleri

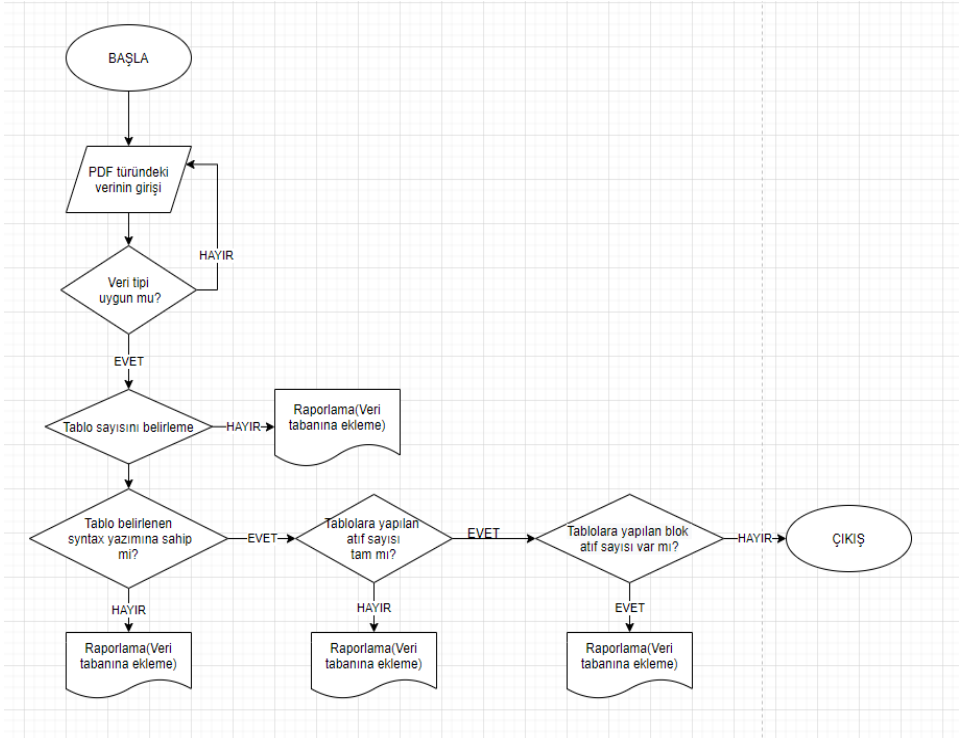


Şekil 4.3.2.3.1 Tablolar Modülü Kullanıcı Arabirimi

Modül Tanımı

Bu modül girdinin tablolar sayısını, tablolar syntax yapısını, blok atıf sayısını ve atıf sayısını kontrol eder.

Modül İç Tasarımı

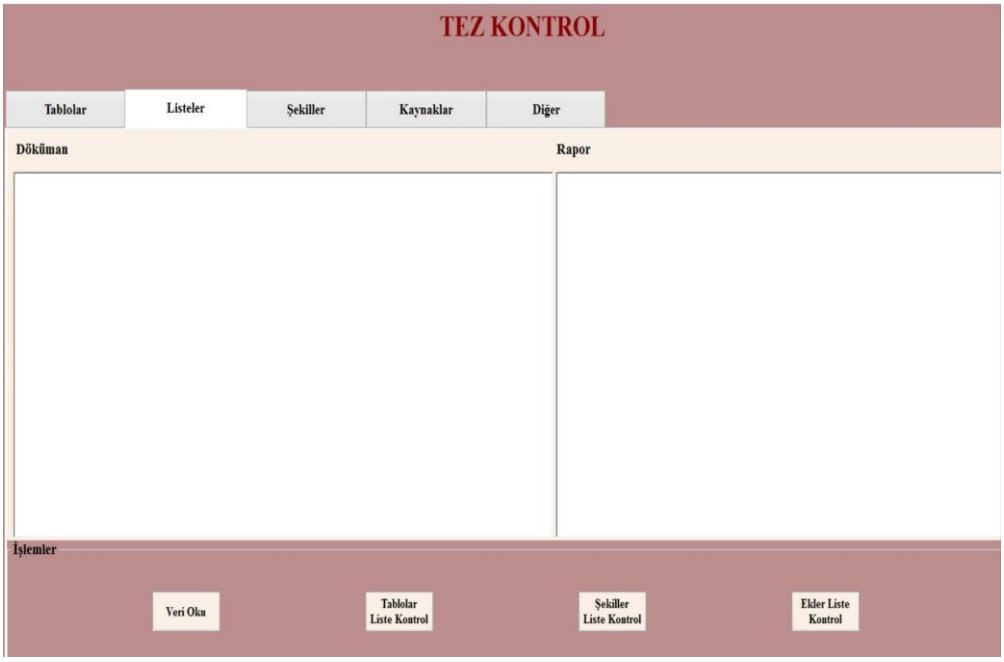


Şekil 4.3.1.3 Tablolar modülü akış diyagramı

4.3.2.4 Listeler Modülü

İşlev

Kullanıcı tarafından yapılan girdinin listeler kontrol işlemlerini gerçekleştirir.



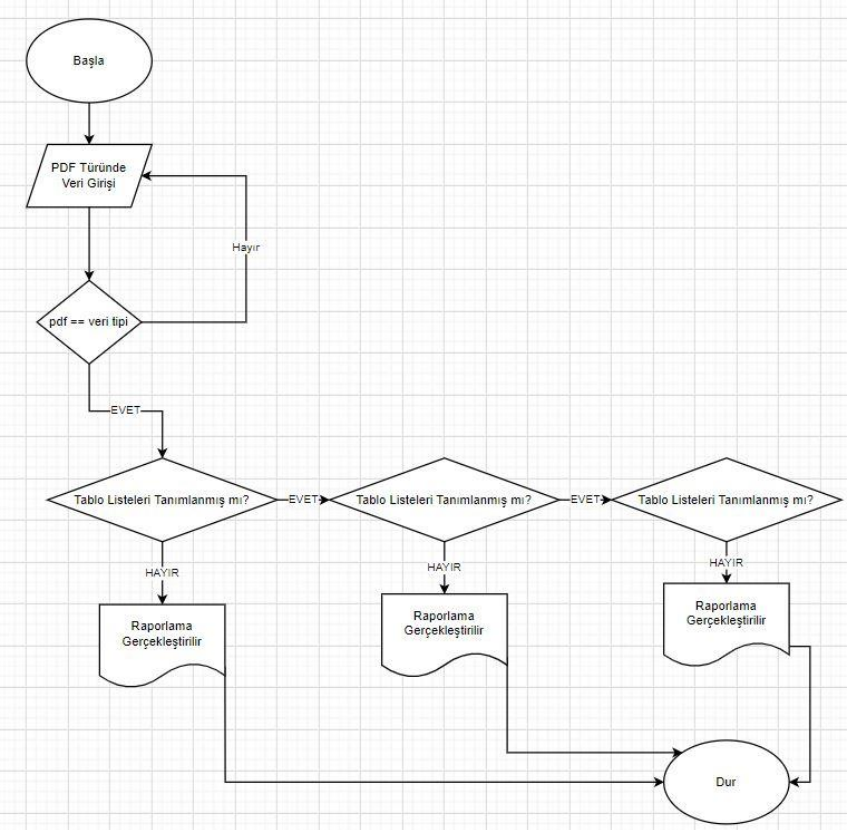
Şekil 4.3.2.4.1 Listeler Modülü Arabirimi

Kullanıcı Arabirimleri

Modül Tanımı

Bu modül girdinin tablolar listesi, şekiller listesi ve ekler listesi başlıklarını ve içindekiler kısmını denetler.

Modül İç Tasarımı



Şekil 4.3.1.5 Listeler modülü akış diyagramı

4.3.3 Kullanıcı Profilleri

Kullanıcı: Veri girdisini yapar. Rapor sonuçlarını alır.

4.3.4 Entegrasyon ve Test Gereksinimleri

Sistemin herhangi bir entegrasyon gereksinimi bulunmamaktadır.

4.4 Ortak Alt Sistemlerin Tasarımı

4.4.1 Ortak Alt Sistemler

Bu sistemde ortak bir alt sistem kullanılmamaktadır.

4.4.2 Modüller Arası Ortak Veriler

Modüller arası ortak veriler kullanılmamaktadır.

4.4.3 Ortak Veriler İçin Veri Giriş ve Raporlama Modülleri

Ortak veriler olmadığı için kullanılmamaktadır.

4.4.4 Güvenlik Altsistemi

Mevcut alt sistemin herhangi bir güvenlik gereksinimi bulunmamaktadır. Bunu sebebi girilen verilerin herhangi bir üçüncü kişiye aktarılmaması ve internet ortamında paylaşılmamasıdır. Burada güvenlik sadece kullanıcı tarafından sağlanabilir.

4.4.5 Veri Dağıtım Altsistemi

Veri dağıtım sistemi bulunmamaktadır. Veri tabanına kayıt edilen veriler kullanıcının kendi bilgisayarında bulunmaktadır. Veri dağıtımının sorumlusu bizzat kullanıcının kendisidir.

4.4.6 Yedekleme ve Arşivleme İşlemleri

Herhangi bir yedekleme ve arşivleme işlemi bulunmamaktadır. Bunların tamamı kullanıcının kendi inisiyatifine bağlıdır.

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.

Şekil 5.1 Gerçekleştirim Modeli

5.2 Yazılım Geliştirme Ortamları

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

- Programlama Dili
- Veri Tabanı Yönetim Sistemi

CASE Araçları belirlendi ve yazılım geliştirme ortamı hazırlandı.

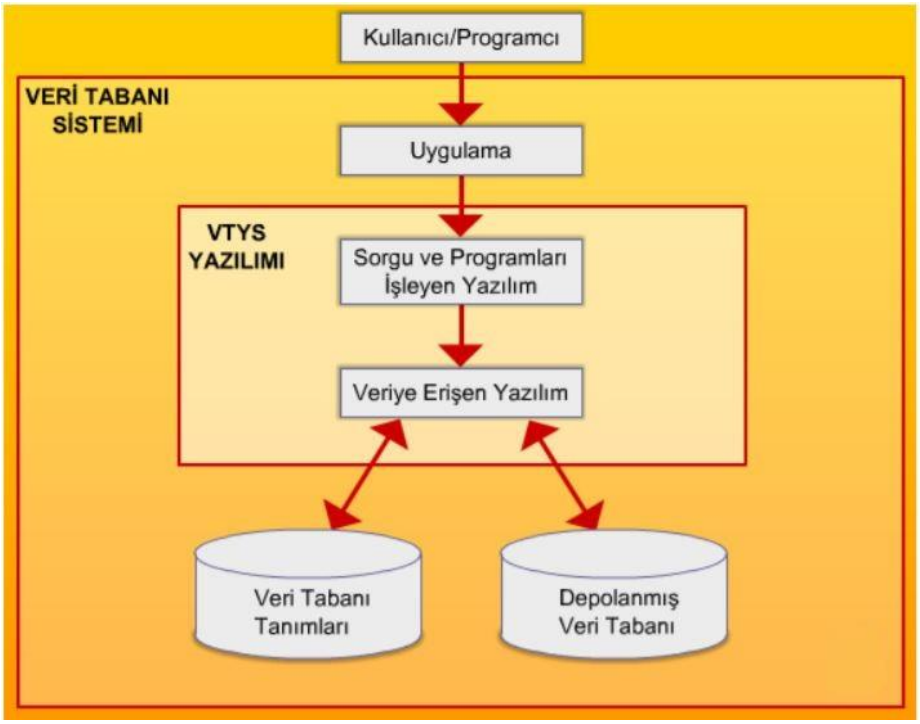
5.2.1 Programlama Dilleri

Sistemin programlanması için kullanacağımız dil C# programlama dilidir. Veri tabanı işlemleri için kullanacağımız dil ise SQL ‘dir.

5.2.2 Veri Tabanı Yönetim Sistemleri

Veri tabanı yönetim sistemi; yeni bir veri tabanı oluşturmak, veri tabanını düzenlemek, geliştirmek ve bakımını yaptırmak gibi çeşitli karmaşık işlemlerin gerçekleştirildiği birden fazla programdan oluşmuş bir yazılım sistemidir. Veri tabanı yönetim sistemi, kullanıcı ile veri tabanı arasında arabirim oluşturur ve veri tabanına her türlü erişimi sağlar.

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



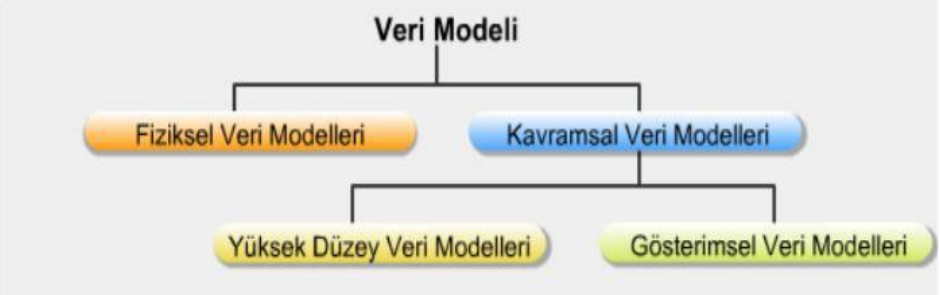
Şekil 5.2.2.1 Veri Tabanı Sistemi

5.2.2.1 VTYS Kullanmanın Ek Yararları

1	Genişleme Potansiyeli
2	Esneklik
3	Uygulama geliştirme zamanının azalması
4	Güncel bilgilerin tüm kullanıcılara aynı zamanda ulaşması
5	Ölçümde ekonomi
6	Ortak verilerin tekrarının önlenmesi ve tutarlılığının sağlanması
7	Her kullanıcıya yalnız ilgilendiği verilerin anlaşılır yapılarda sunulması
8	Uygulama yazılımı geliştirmeyi kolaylaştırması

Şekil 5.2.2.1.1 VTYS yararları

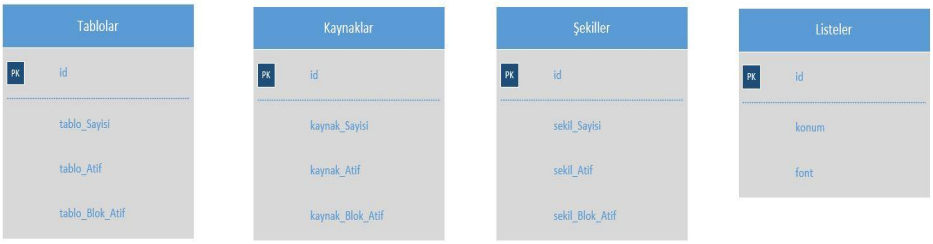
5.2.2.2 Veri Modelleri



Şekil 5.2.2.2.1 Veri Modeli

Fiziksel Veri Modelinde verilerin MSSQL 'de tablolar içindeki alanlarda saklanacağı ve birbirleriyle ilişki içinde olduğunu söyleyebiliriz.

5.2.2.3 Şemalar



Şekil 5.2.2.3.1 Tablo Şemaları

5.2.2.4 VTYS Mimarisi

- **İçsel Düzey**
  Veri tabanının fiziksel saklanma yapısını açıklar. Fiziksel veri modeli kullanır ve veri tabanına erişim yolu ile veri saklamanın tüm detaylarını açıklar.
- **Kavramsal Düzey**
  Kavramsal düzey ise kavramsal şema içerir ve kullanıcılar için veri tabanının yapısını açıklar. Gerçek fiziksel yapının detaylarını kullanıcıdan gizler, veri tipleri, varlıklar ilişkiler, kullanıcı işlemleri ve sınırlamalar üzerine konsantre olmamızı sağlar. Daha yüksek seviyede veri modeli veya gerçekleştirim veri modeli bu seviyede kullanılabilir.
- **Dışsal Düzey**
  Bu düzey, dış şemalar veya kullanıcı görüşleri içerir. Her dış şema veri tabanının bir bölümünü açıklar ve her gruba kendi ilgilendiği görüşü sunarken, diğer bir gruptan ilgilenmediği görüşü saklar. Daha yüksek düzeyde veri modeli veya gerçekleştirim veri, modeli bu seviyede kullanılabilir.

5.2.2.5 Veri Dilleri ve Arabirimleri

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

Sistemimizde veri tabanı dili olarak SQL kullanıldı. Henüz prototip aşamasında olan sistemimiz için MSSQL arabirimini yeterli gördük.

5.2.2.6 Veri Tabanı Sistem Ortamı

Tüm Yükleme, yedekleme, performans ölçme, sıralama, veri sıkıştırma, ve benzeri fonksiyonları yerine getirmek için SQL Server Management System kullandık.

5.2.2.7 VTYS 'nin Sınıflandırılması

Veri modellemesi yapmak amacıyla farklı durumlara uygun olan ve birbiriyle farklı özellikler taşıyan pek çok veri modeli vardır. Veri modelleri aşağıdaki gibi sınıflandırılabilir:

- Basit Veri Modelleri
- Hiyerarşik Veri Modelleri
- Şebeke Veri Modelleri
- Geliştirilmiş Veri Modelleri
- Varlık-İlişki Veri Modelleri (Vİ Modeli)
- İlişkisel Veri Modelleri
- Nesne Yönelimli Veri Modelleri

Kullandığımız veri modelleri aşağıda verilmiştir;

Basit Veri Modelleri

Bilgisayarlarda veri işleme ihtiyacının ortaya çıkmasıyla, dosyalama sistemleri oluşturmak amacıyla kullanılmaya başlanan Hiyerarşik ve Şebeke veri modelleridir.

Hiyerarşik Veri Modelleri

Çoklu ilişkileri temsil edebilmek için, varlık tiplerinin gereksiz veri tekrarı yapmadan her ilişki için ayrı ayrı tanımlanmasına Hiyerarşik veri modeli denilir.

Şebeke Veri Modelleri

Tablo ve grafiklerden oluşan veri modeli Şebeke Veri Modelidir. Kayıt tipi ve bağlantı olmak üzere iki ayrı veri yapılandırma aracı vardır. Kayıt tipleri varlık tiplerini, bağlantılar ise ilişki tiplerini belirler. Bu yapıyı gösteren grafiğe de veri yapısı grafiği adı verilir.

5.2.2.8 Hazır Program Kütüphane Dosyaları

Sistem içerisinde ayrıca eklenilen VTYS hazır programları ya da kütüphane dosyaları bulunmamaktadır. Varsayılan programlar ve kütüphaneler kullanılmıştır.

5.2.2.9 CASE Araç ve Ortamları

CASE araçları olarak Microsoft Visio ve online akış diyagramı programları kullanılmıştır.

5.3 Kodlama Stili

SOLID kurallarına olabildiğince bağlı kalmaya çalışılmıştır. Camel Case standardı ile tanımlamalar yapılmıştır. Bütün komponentler bulunduğu sekmenin özelliklerini genelden özele doğru şekilde almıştır.

5.3.1 Açıklama Satırları

Program içerisinde gerekli kısımlara açıklama satırları yerleştirilmiştir.

5.3.2 Kod Biçimlemesi

Metot kullanımına önem gösterildi. Kodların modüler halinde olması önem arz etmektedir. Region yönetimi kullanıldı.

5.3.3 Anlamlı İsimlendirme

Nesneler kullanım amaçlarına uygun olarak isimlendirilmiştir.

5.3.4 Yapısal Programlama Yapıları

Genel olarak 3 başlıkta incelenirse:

- **Ardışık işlem yapıları**: Bu tür yapılarda genellikle fonksiyon, altprogram ve buna benzer tekrarlı yapılar tek seferde çözüldü.

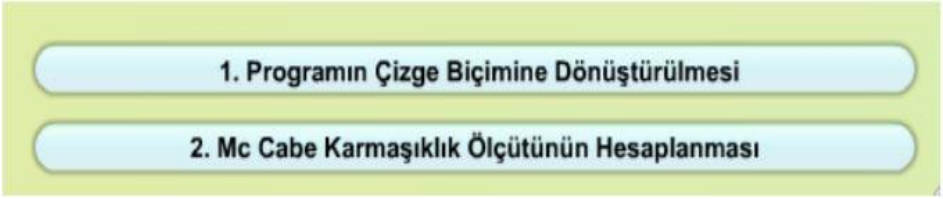
- **Koşullu işlem yapıları**: Bu yapılar ise neredeyse programın tamamında kullanıldı, karşılaştırma yapılan her yerde bu koşullu işlem yapılarına yer verildi.

- **Döngü yapıları**: Tıpkı ardışık işlemler gibi alt alta birkaç satır yazmak yerine tek bir döngüyle bu sorunların üstesinden gelindi.

5.4 Program Karmaşıklığı

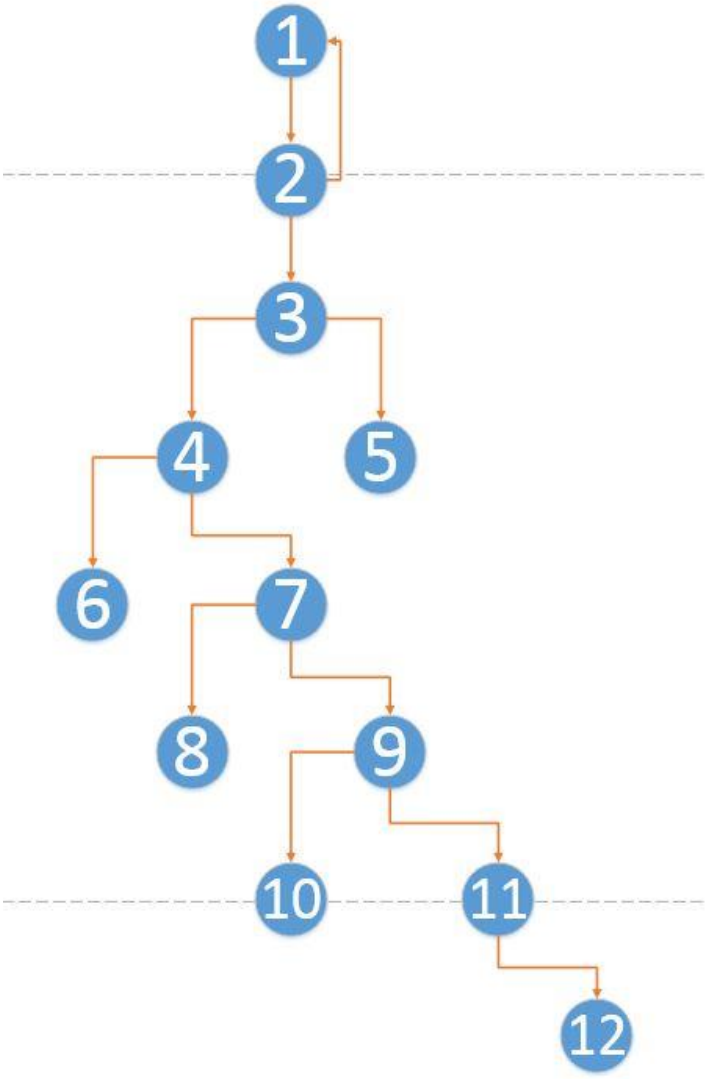
Program karmaşıklığını ölçmek için bir çok teorik model geliştirilmiştir. Bu modellerin en eskisi ve yol göstericisi McCabe karmaşıklık ölçütüdür. Söz konusu ölçüt 1976 yılında McCabe tarafından geliştirilmiştir. Bu konuda geliştirilen diğer ölçütlerin çoğu, bu ölçütten esinlenmiştir.

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



Şekil 5.4.1 Program karmaşıklığı

5.4.1 Programın Çizge Biçimine Dönüştürülmesi



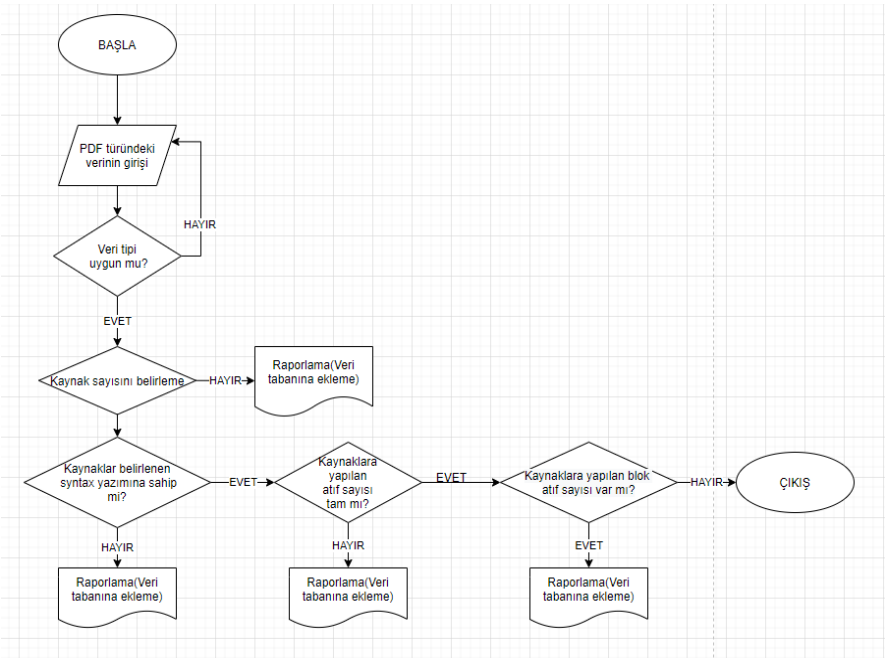
Şekil 5.4.1.1 Programın Kaynaklar Modülünün Çizgi Hali

5.4.2 McCabe Karmaşıklık Ölçütü Hesaplama

V(G) = k – d + 2p

K(Kenar Sayısı)= 12
D(Düğüm Sayısı)= 12
P(Bileşen sayısı)= 1

Karmaşıklık: 2 ‘dir.



Şekil 5.4.2.1 Kaynaklar Modülü Akış Diyagramı

5.5 Olağan Dışı Durum Çözümleme

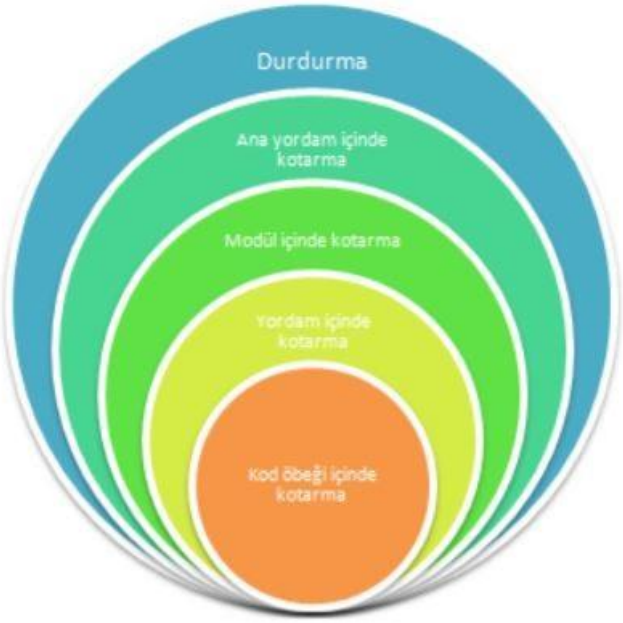
Olağan dışı durum, bir programın çalışmasının, geçersiz ya da yanlış veri oluşumu ya da başka nedenlerle istenmeyen bir biçimde sonlanmasına neden olan durum olarak tanımlanmaktadır.

5.5.1 Olağandışı Durum Tanımları

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

5.5.2 Farklı Olağandışı Durum Çözümleme Yaklaşımları

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



Şekil 5.5.2.1 Olağan dışı halde yapılacaklar

5.6 Kod Gözden Geçirme

Program sınama, 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 girdi olacağını bilerek program yazdığında daha etkin, az hatalı ve okunabilir programlar elde edilebilmektedir.

5.6.1 Gözden Geçirme Sürecinin Düzenlenmesi

Gözden geçirme sürecinin temel özellikleri;

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

5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular

Bir program incelenirken, programın her bir öbeği (yordam ya da işlev) için bazı sorulara yanıtlar aranır. Bu sorulara ek sorular eklenebilir.

5.6.2.1 Öbek Arayüzü

Oluşturulan öbekleri test etmek amacı ile bazı sorular sorulur. Bu sorular:

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

5.6.2.2 Giriş Açıklamaları

Oluşturulan giriş açıklamalarını test etmek amacı ile bazı sorular sorulur. Bu sorular:

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

5.6.2.3 Veri Kullanımı

Oluşturulan veri kullanımını test etmek amacı ile bazı sorular sorulur. Bu sorular:

- İşlevsel olarak ilintili bulunan veri elemanları uygun bir mantıksal veri yapısı içinde gruplanmış mı?
- Değişken adları, işlevlerini yansıtacak biçimde anlamlı mı?
- Değişkenlerin kullanımları arasındaki uzaklık anlamlı mı?
- Her değişken tek bir amaçla mı kullanılıyor?
- Dizin değişkenleri kullanıldıkları dizinin sınırları içerisinde mi tanımlanmış?
- Tanımlanan her gösterge değişkeni için bellek ataması yapılmış mı?

5.6.2.4 Öbeğin Düzenlenişi

- Modüller birleşimi uyumlu mu?
- Modüller arası veri aktarımları sağlanıyor mu?
- Bütün modüller birleştiğinde sistem çalışıyor mu?

Bu sorular gözden geçirme sırasında referans alınacak sorular olacaklardır.

5.6.2.5 Sunuş

Sunuş kısmına gelindiğinde ise şu sorular sorulacaktır:

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

6 Doğrulama ve Geçerleme

6.1 Giriş

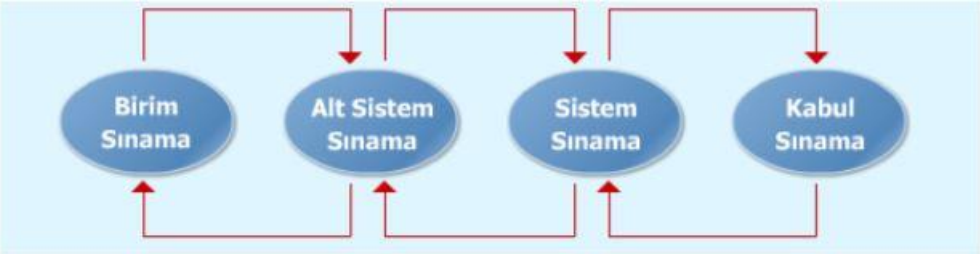
Geliştirilecek bilgi sistemi yazılımının doğrulanması ve geçerlenmesi, üretim süreci boyunca süren etkinliklerden oluşur. Söz konusu etkinlikler:

- Yazılım belirtimlerinin ve proje yaşam sürecindeki her bir etkinlik sonunda alınan çıktıların, tamam, doğru, açık ve önceki belirtimleri 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 belirtimlerin önceki belirtimlerle karşılaştırılması

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

Şekil 6.1.1 Doğrulama ve Geçerleme

6.2 Sınama Kavramları



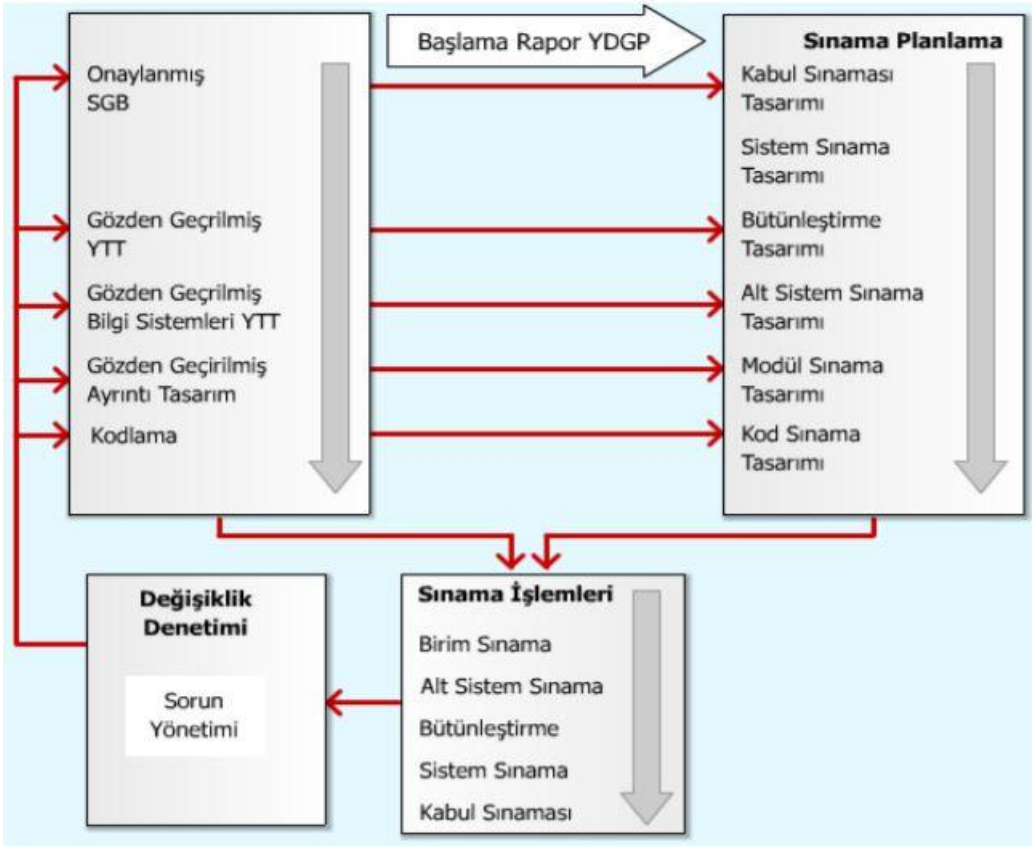
Birim Sınama : Sistemin birimlerinden olan kaynaklar, tablolar, şekiller ve denklemler sırasıyla kendi içlerinde sınandı ve sonuçları çıkartıldı.

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

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

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

6.3 Doğrulama ve Geçerleme Yaşam Döngüsü



Şekil 6.3.1 Yaşam Döngüsü

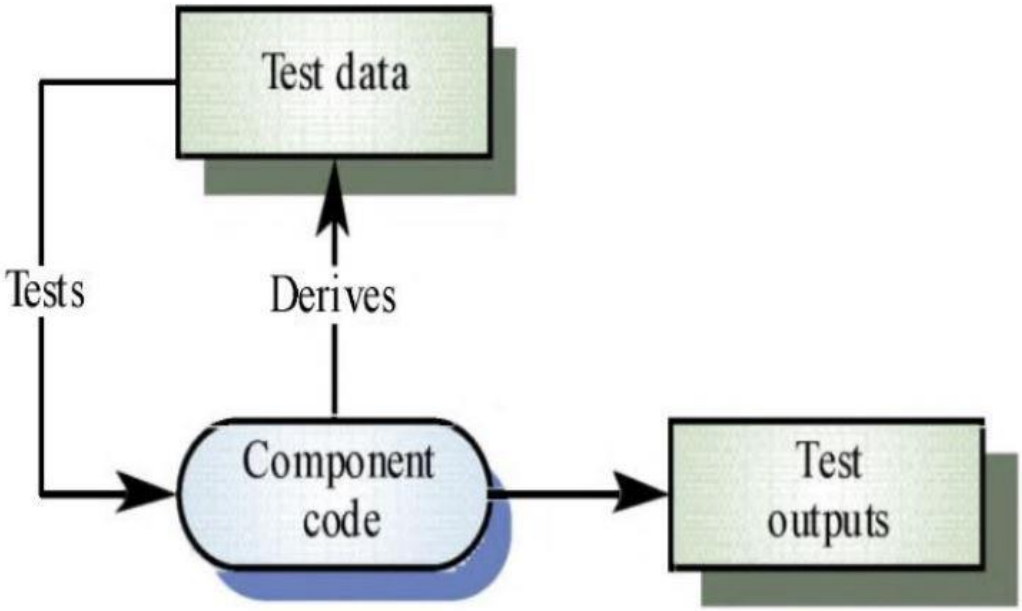
6.4 Sınama Yöntemleri

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

6.4.1 Beyaz Kutu Sınaması

Denetimler arasında:

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

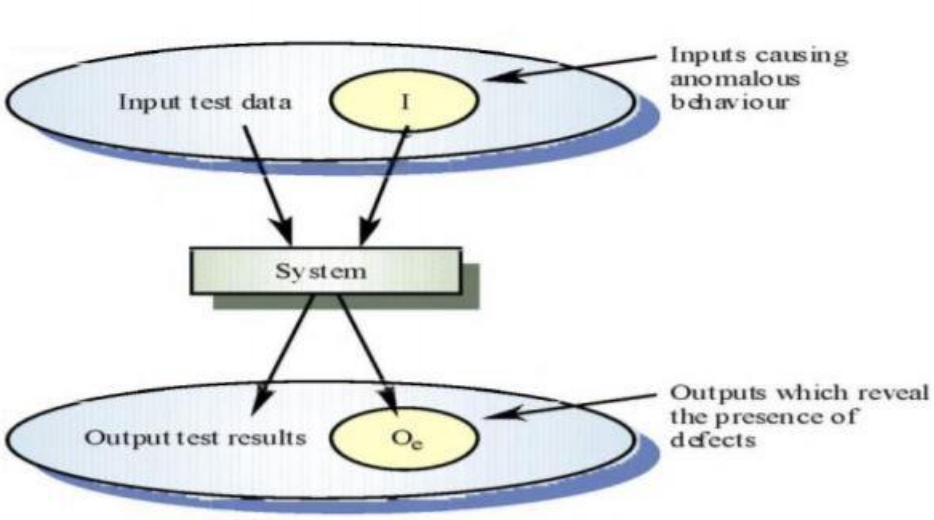


Şekil 6.4.1.1 Beyaz Kutu Sınaması

6.4.2 Temel Yollar Sınaması

Sistemin tümüne yönelik işlevlerin doğru yürütüldüğünün kontrol testidir. Sistemin şartnamesini ve gereklerini inceler.

- Eş değerlere bölme
- Uç değerler analizi
- Karar tablosu
- Sonlu durum makinesi
- Belgelenmiş özelliklere göre test
- Rastgele test
- Kullanım profili



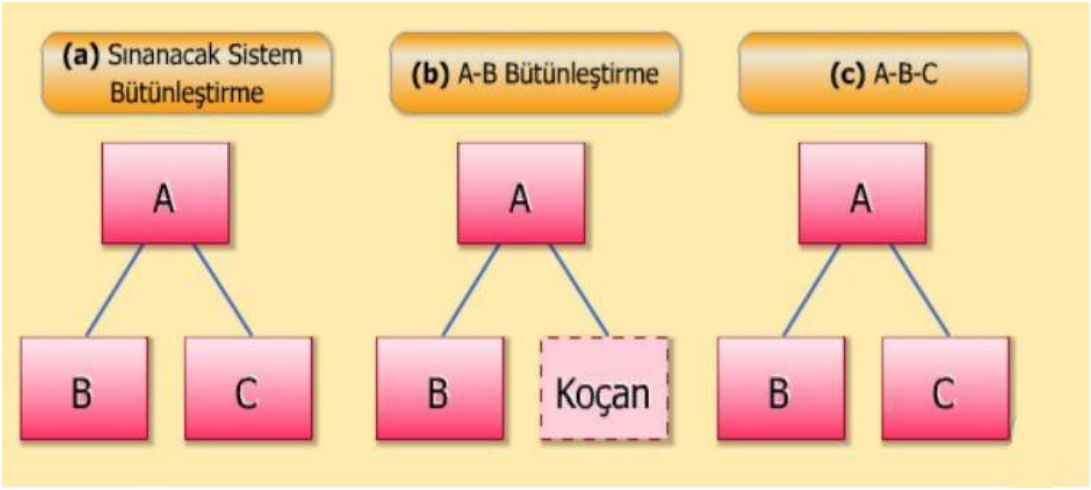
Şekil 6.4.2.1 Sınama şekli

6.5 Sınama ve Bütünleştirme Stratejileri

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

6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme

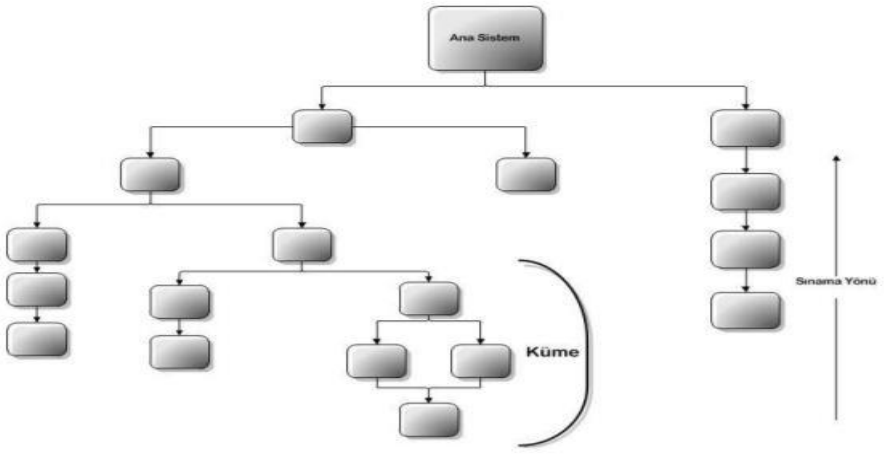
Yukarıdan aşağı bütünleştirmede, önce sistemin en üst düzeylerinin sınanması ve sonra aşağıya doğru olan düzeyleri, ilgili modüllerin takılarak sınanmaları söz konusudur. En üst noktadaki bileşen, bir birim/modül/alt sistem olarak sınandıktan sonra alt düzeye geçilmelidir. Ancak bu en üstteki bileşenin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



Şekil 6.5.1.1 Yukarıdan aşağı sınama

6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme

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



Şekil 6.5.2.1 Aşağıdan yukarı bütünleştirme

6.6 Sınama Planlaması

Bir tablo ile özetlemek gerekirse şu şekilde özetleyebiliriz.
Test raporu hazırlanırken şu özellikler mutlaka planda belirtilmelidir;

Test planı kimliği: Test planının adı veya belge numarası

Giriş: Test edilecek yazılımın elemanlarının genel tanıtım özetleri. Ayrıca bu plan kapsamı ve başvurulan belgeler. Kısaltmalar ve terim açıklamaları bu bölümde bildirilmelidir.

Test edilecek sistem: Sistemde bileşenleri sürüm sayıları olarak sıralar ve sistemin özelliklerini bileşenlerini ve nasıl kullanıldıkları açıklanmalıdır. Ayrıca sistemde test edilmeyecek parçalar belirtilmelidir.

Test edilecek ana fonksiyonlar: Sistemin test edilecek ana fonksiyonlarının kısa bir tanıtımı yapılmalıdır.

Test edilmeyecek ana fonksiyonlar: Sistemde test edilmeyecek fonksiyonları ve bunların neden test edilmedikleri açıklanacaktır.

Geçti/Kaldı Kriterleri: Bir test sonucunda sistemin geçmiş veya kalmış sayılacağını açıklanmalıdır.

Test dokümanı: Test süresince yapılan işlemleri alınan raporları elde edilen bilgileri rapor içinde sunulmalıdır.

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

Riskler ve Önlemler: Test planında varsayılan ve olası yüksek riskli durumları belirtir ve bu durumların olması durumunda, etkilerinin en aza indirilebilmesi için alınması gereken 79 önlemleri açıklar.

6.7 Sınama Belirtimleri

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

Bu ayrıtılar 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 belirtimlerinin hazırlanmasındaki temel maç, etkin sınama yapılması için bir rehber oluşturması gerekir.

Sınama işlemi sonrasında bu belirtimlere,

- sınamayı yapan,
- sınama tarihi,
- bulunan hatalar ve açıklamaları

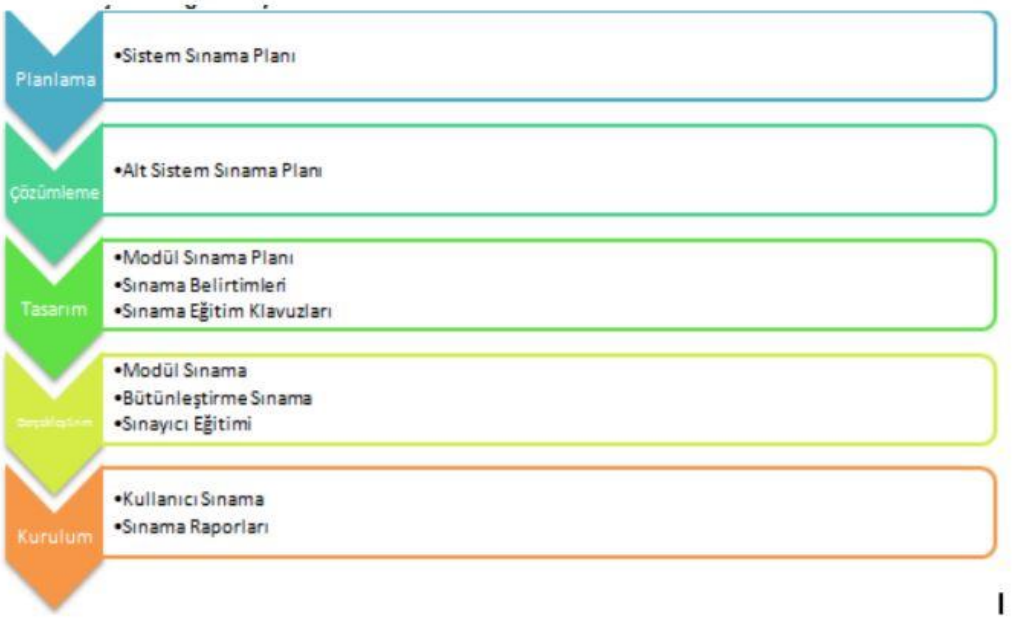
türündeki bilgiler eklenerek sınama raporları oluşturulur.

Sınama raporları, sınama bitiminde imzalanır ve yüklenici ile iş sahibi arasında resmi belge niteliği oluşturur.

6.8 Yaşam Döngüsü Boyunca Sınama Etkinlikleri

Bütün bu etkinlikleri bir hiyerarşi altında incelemek gerekirse:

- Planlama aşamasında genel planlama sınaması gerçekleştirilir.
- Bu olan tüm planların basit bir ön hazırlığı niteliğindedir.
- Çözümleme aşamasında sınama planı alt sistemler bazında ayrıntılandırılır.
- Tasarım aşamasında sınama plana detaylandırılır ve sınama belirtimleri oluşturulur.
- Bu oluşumlar daha sonra eğitim ve el kitabında kullanılır.
- Gerçekleştirim aşamasında teknik sınamalar yapılır sınama raporları hazırlanır ve elle tutulur ilk testler yapılır.
- Kurulum aşamasında sistemle ilgili son sınamalar yapılır ve sınama raporları hazırlanır.



Şekil 6.8.1 Sınama Etkinlikleri

Bu ayrıtılar temel olarak:

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

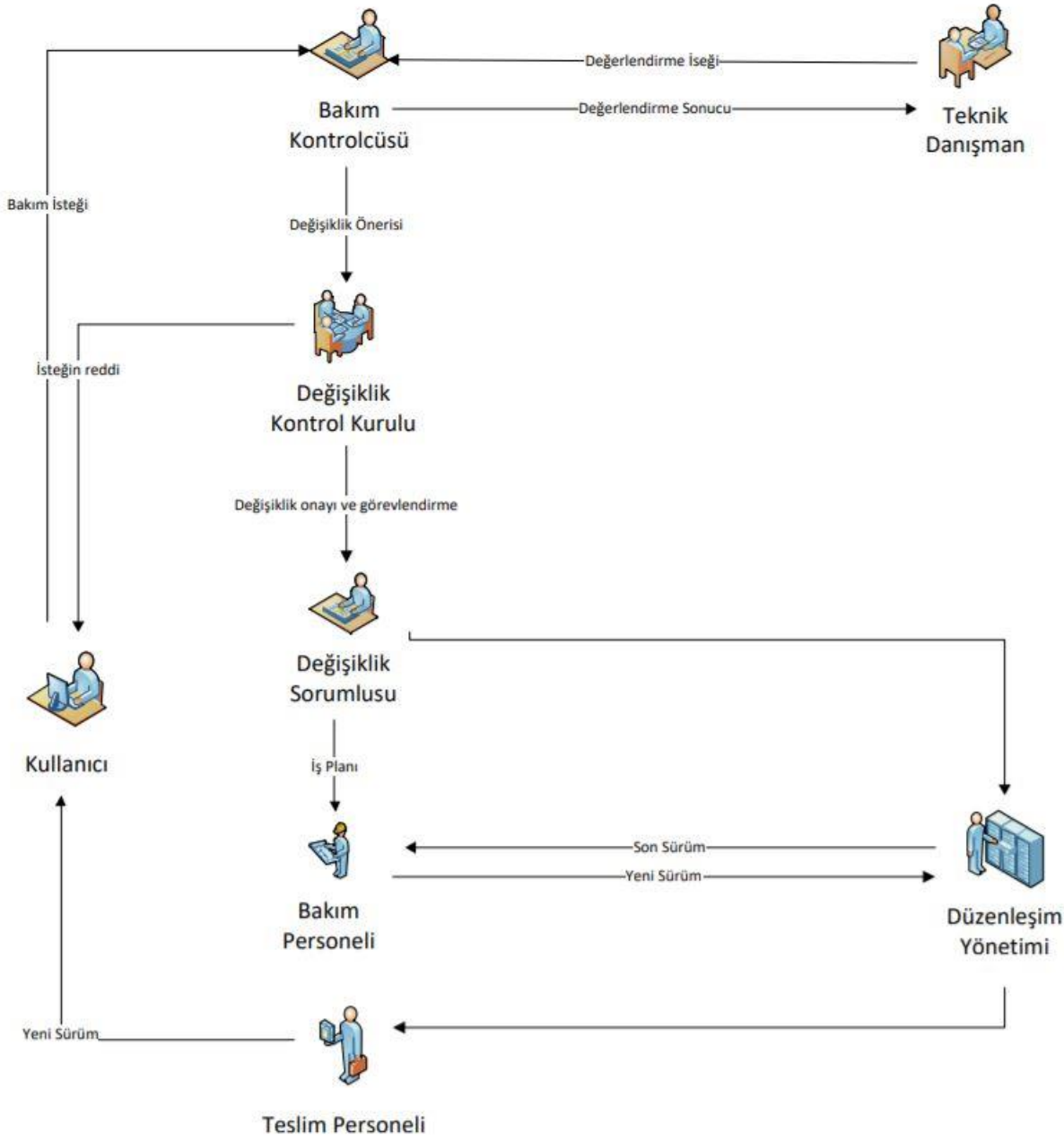
Sınama sırasında bulunan her hata için, **değişiklik kontrol sistemine (DKS),** "Yazılım Değişiklik İsteği" türünde bir kayıt girilir. Hatalar, DKS kayıtlarında aşağıdaki gibi gruplara ayrılabilir:

• **Onulmaz Hatalar:** BT projesinin gidişini bir ya da birden fazla aşama gerileten ya da düzeltilmesi mümkün olmayan hatalardır.
• **Büyük Hatalar:** Projenin kritik yolunu etkileyen ve önemli düzeltme gerektiren hatalardır.
• **Küçük Hatalar:** Projeyi engellemeyen, ve giderilmesi az çaba gerektiren hatalardır.
• **Şekilsel Hatalar:** Heceleme hatası gibi önemsiz hatalardır.

7 Bakım

7.1 Giriş

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



Şekil 7.1.1 Bakım Planı

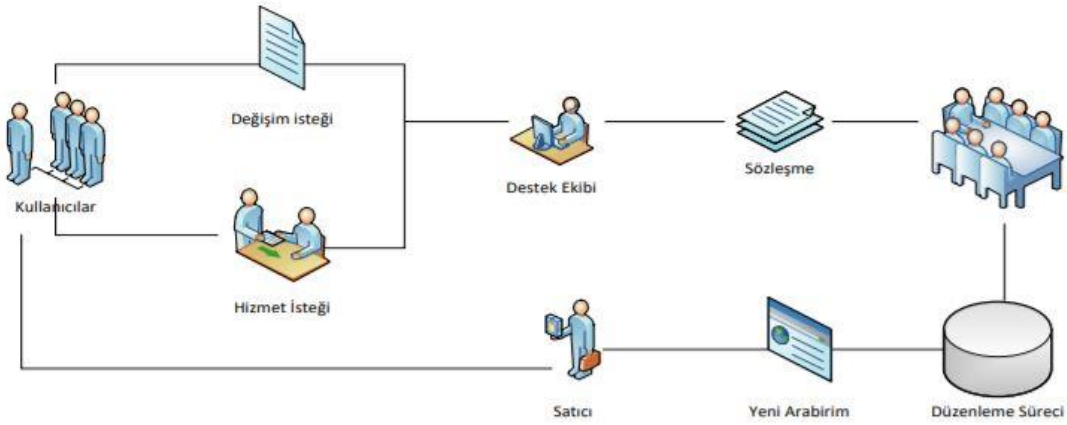
7.2 Kurulum

Sistem kurulumu için indirme dosyası oluşturulup doğrudan "exe." dosyası halinde çalışabilir olacaktır. Kurulumdan kullanıcı sorumludur.

7.3 Yerinde Destek Organizasyonu

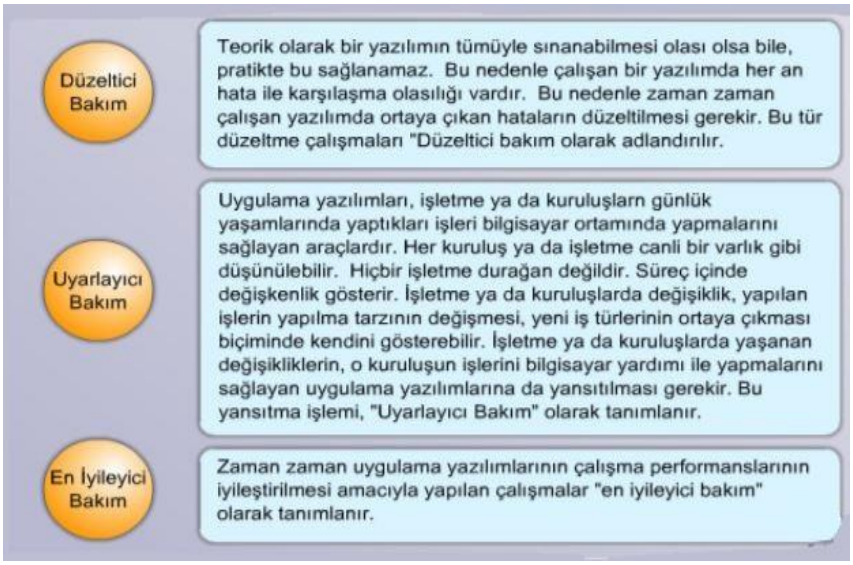
Yerinde destek bulunmamaktadır. Kullanıcı geri dönütleri ile uygulama güncellenecektir.

7.4 Yazılım Bakımı



Şekil 7.4.1 Bakım Planı

7.4.1 Tanım

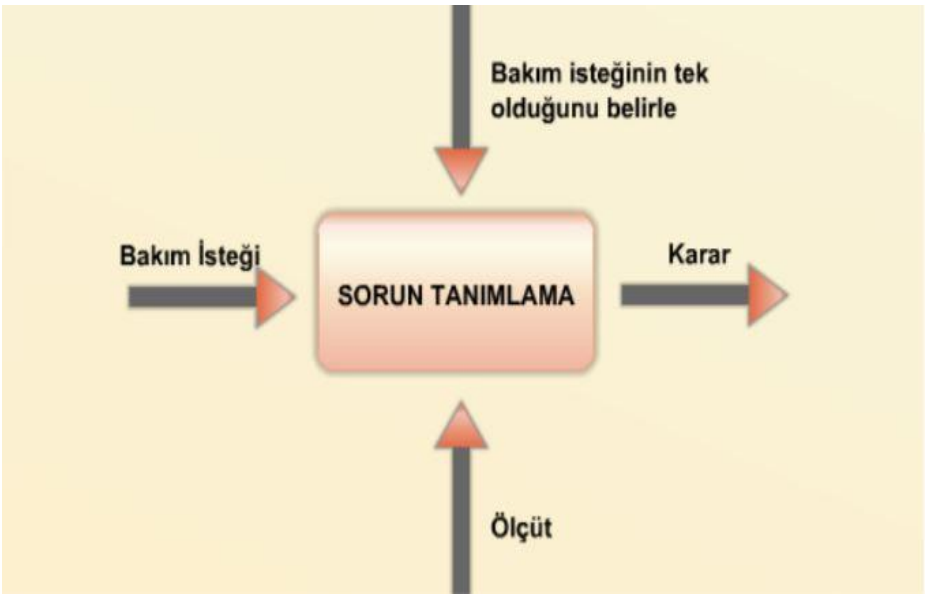


Şekil 7.4.1.1 Bakım çeşitleri

7.4.2 Bakım Süreç Modeli

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

1. Adım: Sorunu Tanımlama Süreci
İlk önce bakım ne için yapılıyor sorun ne buna bir bakalım.



Şekil 7.4.2.1 Sorun Tanımlama

2. Adım: Çözümleme Süreci
Sorun tanımlamadan çıkan karar doğrultusunda problemi kâğıt üzerinde çözelim.

Şekil 7.4.2.2 Çözümleme Süreci

3. Adım: Tasarım Süreci
Çözümlenen sistem sonucunda tasarımı güncelleştirmeye geldi sıra.

Şekil 7.4.2.3 Tasarım Süreci

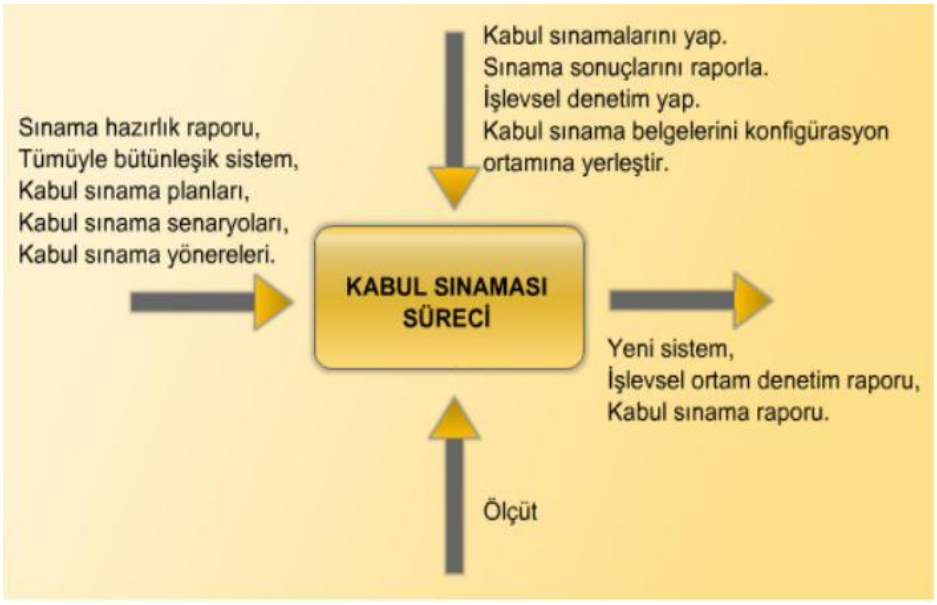
4. Adım: Gerçekleştirim Süreci
Tasarımı yapılan sistemin gerçekleştirmesine sıra geldi.

Şekil 7.4.2.4 Gerçekleştirim Süreci

5. Adım: Sistem Sınama Süreci
Artık tekrardan tasarlanan sistemin sınama sürecini tekrar ele almak gerekiyor.

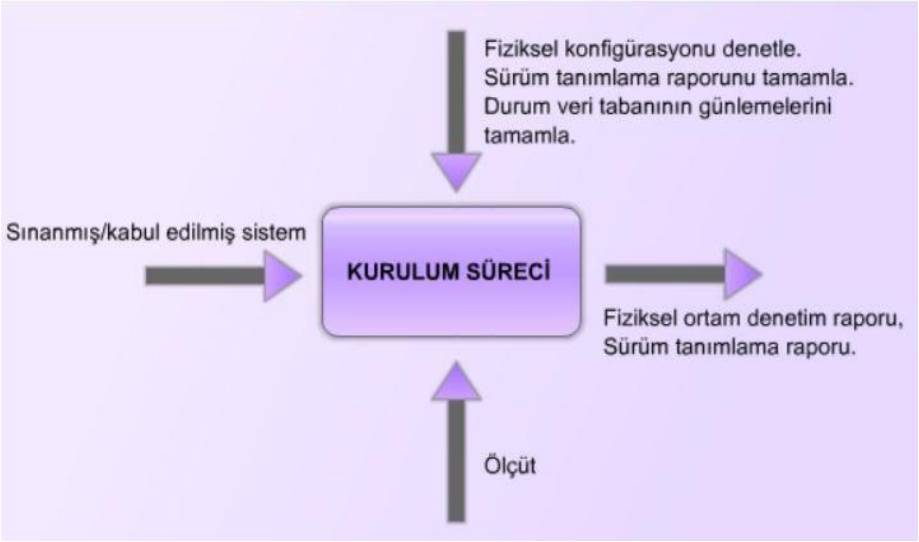
Şekil 7.4.2.5 Sınama Süreci

6. Adım: Kabul Sınaması Süreci Kendi içimizde sınadığımız sistemi birde müşteri karşısında sınıyoruz.



Şekil 7.4.2.6 Kabul Sınaması

7. Adım: Kurulum Süreci Kabul sınamasını geçen sistemimiz artık tekrardan kurulum aşamasına geçiyor.



Şekil 7.4.2.7 Kurulum Süreci

8 Sonuç

Sonuç olarak sistem hayata geçirildiğinde ne gibi avantajlar sağlayacağı belirtilmiştir. Akademik camiaya kazandırılmak istenen makalelerin daha hızlı, güvenilir ve doğru bir şekilde kontrollerinin gerçekleştirilmesi sağlanmış olacaktır.

9 Kaynaklar

[1] http://fbe.firat.edu.tr/sites/fbe.firat.edu.tr/files/FU_FBE_%C3%96rnek_Tez_O_Orhan_YL_Parametresiz.pdf