T.C.
FIRAT ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

Proje Dokümantasyonu

Tez Kontrol Sistemi

Proje Ekibi

15542021 Murat Gökce

15542012 Mehmet Ali Şimşek

OCAK-2021

ÖNSÖZ

Bu projede tez çalışmalarının belirtilen kurallara uyumluluğu ve içerdiği bilgileri destekliyor mu gibi tezlerin doğru yapılması için tasarlanmıştır. Projenin analizinde, tasarımında, gerçekleştiriminde, testinde ve hayata geçirilmesinde desteği olan çalışma arkadaşlarımıza ve bu projenin geliştirilmesinde yardımcı olan Doç. Dr. Fatih ÖZKAYNAK'A sonsuz teşekkürlerimizi iletiriz.

1. GİRİŞ

1.1. Projenin Amacı

Şüphesiz ki üniversitelerde tez hazırlamak büyük bir konu. Bu konuda çoğu zaman zor süreçler yaşanabiliyor. Yapmak istediğimiz sistemle birlikte hem zor süreçleri ortadan kaldırmak istiyoruz hem de tez yazım kurallarına uygunluk sağlandı mı, tezde yararlanılan verilerin gerçek verilerle ilişkisini, yararlanılan kaynakların tezde kullanımını kontrol eden, eğer şekil ya da tablo kullanıldıysa sistemde doğru kullanılmış mı, tezin savunma zamanı intihal olup olmadığını sağlayan bir proje hedeflemekteyiz.

1.2 Projenin Kapsamı

Her alanda ki tezde rahatça kullanılabilecek ve herkesin kolay bir şekilde uzun vadede kullanabileceği bir sistemdir. Tezin savunma zamanında ise Kullanım alanları;

- Tezi yazan kişiler
- Tezin sunulduğu makamlar
- Üniversiteler

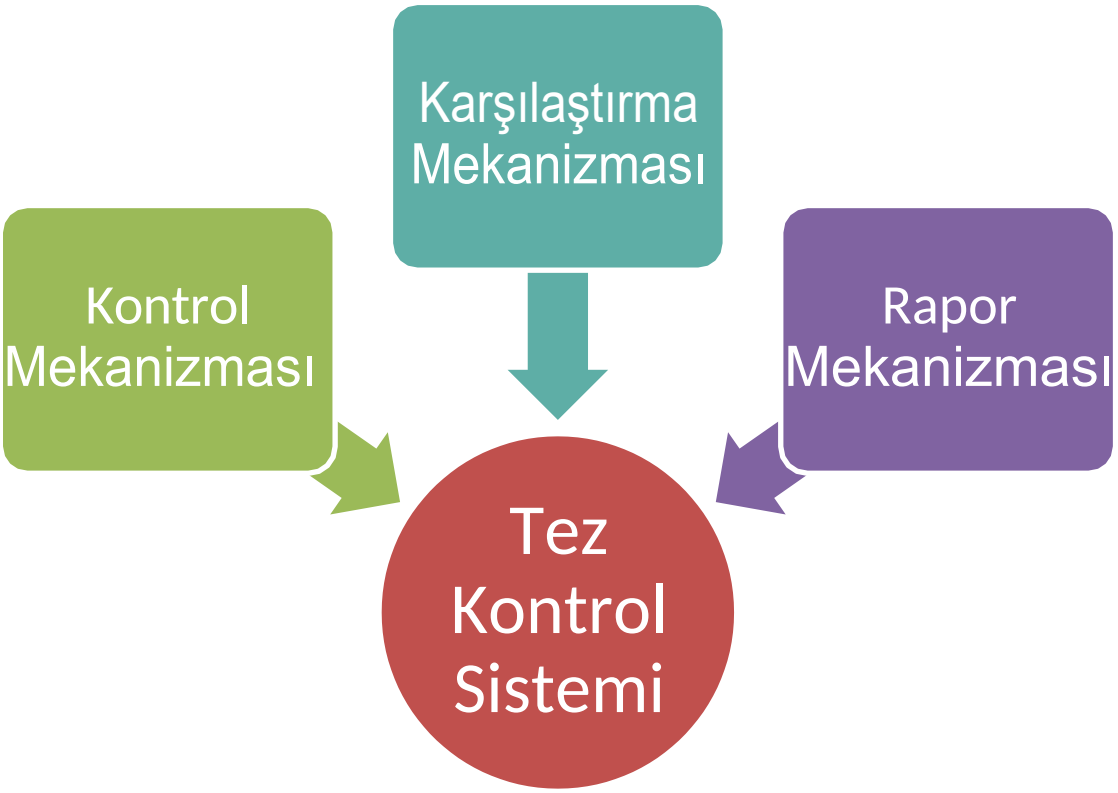
1.3 Tanımlamalar ve Kısaltmalar

-TKS:Tez Kontrol Sistemi

2. PROJE PLANI

2.1 Giriş

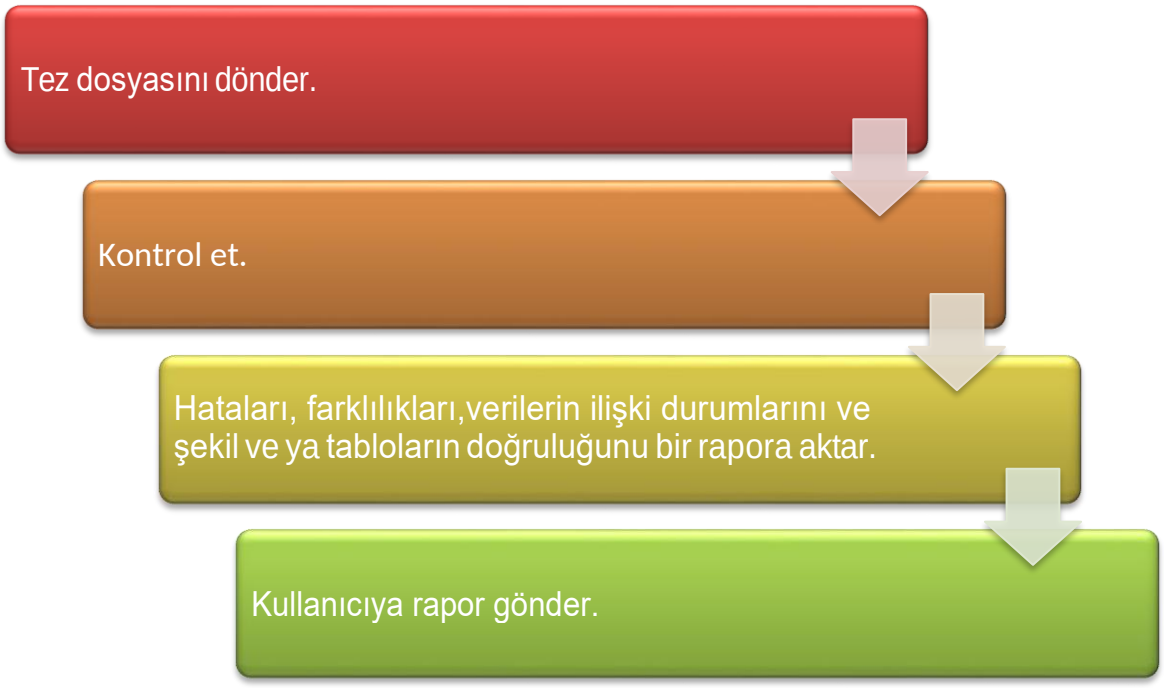
Sav veya tez, bilimsel yöntemde belli ön bilgilere dayanılarak, henüz kanıtlanmamış fakat mevcut bilgilerle mantıksal olarak çelişmeyen, bilimsel araştırma sürecinde doğrulanmaya çalışılan düşüncelerdir. Tez çalışması yapıldıktan sonra intihal işleminden geçmesi gerekir, bundan sonra tez savunması yapılabilir. Bununla da, lisans derecesi veya yüksek lisans diploması ve derecesine sahip olunur. Oldukça uzun bir süreye dayanan tez yazımı, belirli kurallar etrafında şekillendirilerek ilerlemeye dayanan bir süreçtir



Şekil 2.1 Projenin Genel Yapısı

2.2 Projenin Plan Kapsamı

Projenin plan kapsamında genel olarak mevcut sistem, sistemin gerekliliği ve bu sistemin güvenilirliğinden yola çıkıldı. Bu sistemle birlikte tüm yazım hatalarını engellemek, intihal olup olmadığını saptamak, verilerin doğruluğunu ilişkilendirmek ve rapor alarak tezin konusuna uygunluğunu kontrol ederek zaman kaybı olmadan kontrol edilebilecektir.



Şekil 2.1 Projenin Genel işlevi

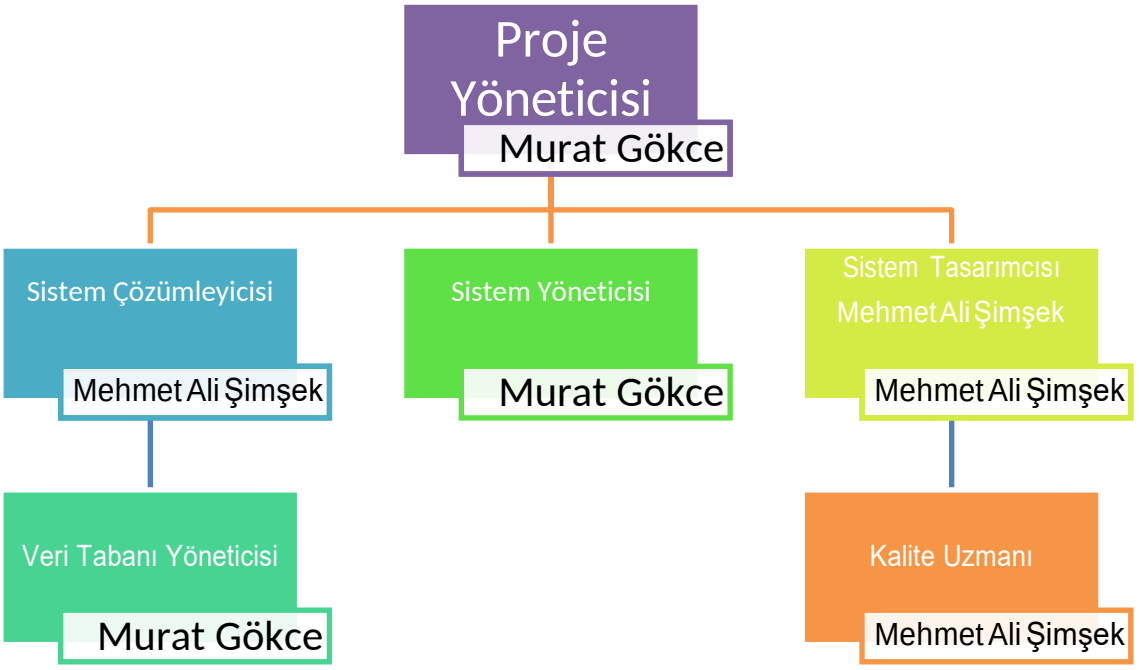
Bu sistem çoğu üniversite ve kullanıcı tarafından desteklenecektir. Bunun nedeni ise zaman ve maliyet kazancı olacaktır.

2.3 Proje Zaman-İş Planı

Zaman-İş Planı									
İş-Zaman	**1.Hafta**	**2.Hafta**	**3.Hafta**	**4.Hafta**	**5.Hafta**	**6.Hafta**	**7.Hafta**	**8.Hafta**	**9.Hafta**
Proje Planı	✓								
Analiz		◷	✓						
Sistem Çözümleme			◷	✓					
Sistem Tasarımı				◷	✓				
Gerçekleştirim					◷	◷	✓		
Test							◷	✓	
Sunum									✓

Şekil 2.2 Projenin Zaman-İş Planı

2.4 Proje Ekip Yapısı

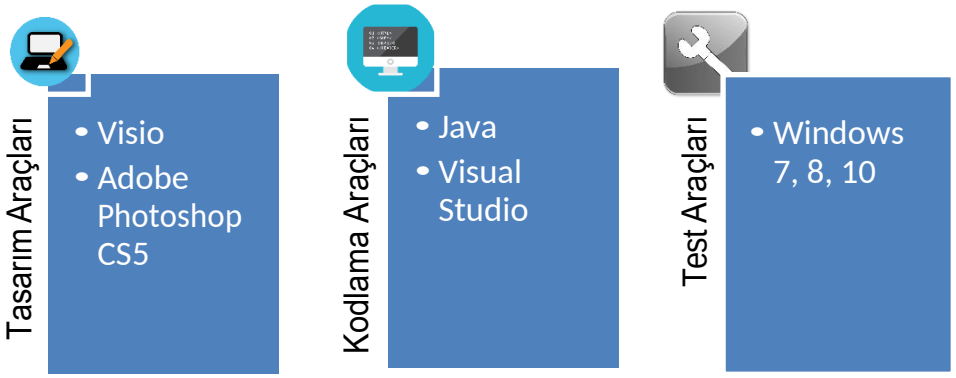


Şekil 2.3 Projenin Ekip Planı

2.5 Önerilen Sistemin Teknik Tanımları

- Karşılaştırma kuralları
- Syntax ve semantik yapılar

2.6 Kullanılan Özel Geliştirme Araçları



Şekil 2.4 Projenin Geliştirme Araçları

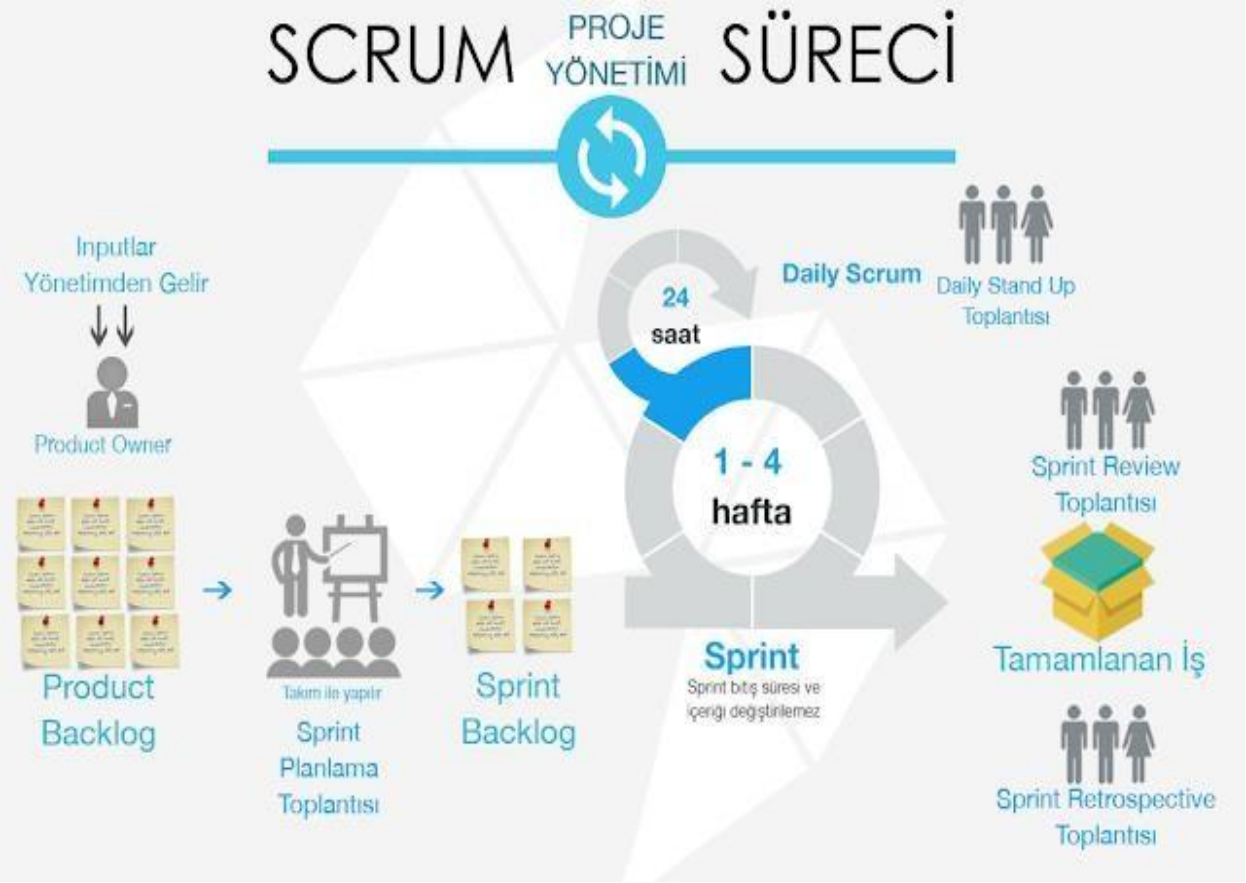
2.7 Proje Standartları, Yöntem ve Metodolojiler

Scrum; Agile proje yönetim metodolojilerinden biridir. Kompleks yazılım süreçlerinin yönetilmesi için kullanılır. Bunu yaparken bütünü parçalayan; tekrara dayalı bir yöntem izler. Düzenli geri bildirim ve

planlamalarla hedefe ulaşmayı sağlar. Bu anlamda ihtiyaca yönelik ve esnek bir yapısı vardır. Müşteri ihtiyacına göre şekillendiği için müşterinin geri bildirimine göre yapılanmayı sağlar. İletişim ve takım çalışması çok önemlidir. 3 temel prensip üzerine kurulmuştur;

- **Şeffaflık;** Projenin ilerleyişi, sorunlar,gelişmeler herkes tarafından görülebilir olmalıdır.
- **Denetleme;** Projenin ilerleyişi düzenli olarak kontrol edilir.
- **Uyarlama;** Proje, yapılabilecek değişikliklere uyum sağlayabilmelidir.

Bu nedenle projede scrum yöntemi kullanılmıştır.



Şekil 2.5 Scrum Süreci

2.8 Kalite Sağlama Planı

Şekil 2.6 Kalite Sağlama Planı

Projedeki kalite sağlama planımız yukardaki tabloda da belli olduğu üzere;

1. **Ekonomi:** Ekonomik açıdan yazılımın maliyeti her ne kadar ilk seferde pahalı olsa da ileriye dönük düşünüldüğünde ve zaman tasarrufundan ötürü gayet uygundur.
2. **Tamlık**: Projede herhangi bir açık olmamalı ve programda bulunan tüm butonlar textler vs. çalışır ve tamdır.
3. **Yeniden Kullanılabilirlik**: Otomasyon her koşulda tekrardan düzenlenip kullanılabilecek.
4. **Etkinlik**: Kullanıcı sistemin her alanına hakim olduğu için sistemi etkin bir biçimde kullanacak.
5. **Bütünlük**: Admin1 sistemin tüm kısımlarına hakim olacak ve program bir bütün halinde çalışacaktır.
6. **Güvenilirlik**: Otomasyon gerekli güvenlik önlemlerinin alınması yanı sıra şuan devlet bünyesinde bulunan çok yüksek güvenlik önlemli serverlarda saklanacaktır.
7. **Modülerlik**: Modülerlik otomasyonun her seviyesindeki kişinin ayrı ayrı sayfalardan söz sahibi olmasını sağlar. Örneğin: Yönetim modülü, Giriş Modülü…
8. **Belgeleme**: Bu belgeden de anlaşılacağı üzere tam anlamıyla sistemin özeti olacak bu

doküman oluşturulmuştur.

9. **Kullanılabilirlik**: Kullanılabilirlik olarak her seviyedeki insana hitap edeceğinden zor renkler karmaşık sistemlerden kaçınılmıştır.

10. **Temizlik**:

11. **Değiştirilebilirlik**: Otomasyonun veri tabanını erişme yetkisi olan ve sistem hakkında bilgisi olan herkes sistemde değişiklik yapabilecek.

12. **Esneklik**: Proje farklı platformlarda ve internet üzerinden çalışacağından gayet esnektir.

13. **Genellik**: Proje her üniversitede kullanılabileceğinden geneldir. Ve Türkiye genelinde kullanılacaktır.

14. **Sınanabilirlik**: Projedeki pilot bölge uygulaması sınana bilirliğinin göstergesidir.

15. **Taşınabilirlik**: Sistem internet üzerinden kullanılacağından herhangi bir özel cihaz gerektirmez ve istenilen cihazlarda taşınabilir ve kullanılabilir.

16. **Birlikte Çalışılabilirlik:** Bu projedeki en büyük sıkıntı olacak veri girişi şuanda var olan ve her bireyin bilgilerinin saklayan sistemle birleşik ve eş zamanlı çalışmakta.

2.9 Konfigürasyon Yönetim Planı

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

- Üniversiteye giren yeni akademik görevli için sistem üye kadrosu açması,
- Herhangi bir sebepten dolayı ayrılan personel olması,
- Sistemde herhangi bir istenmeyen durum

halinde, Durumları için konfigürasyon yönetim planı

oluşturuldu.

2.10 Kaynak Yönetim Planı

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

2.11 Eğitim Planı

Projeden kazanılacak en önemli olaylardan biride eğitimdir. Kullanılacak dillerin ara yüz editör ve programların kullanımında hakim olunamaması halinde bu program başarıyla neticelendirilemez. Ancak projede kullanılan programlara hakim olunduğundan dolayı eğitime gerek yoktur.

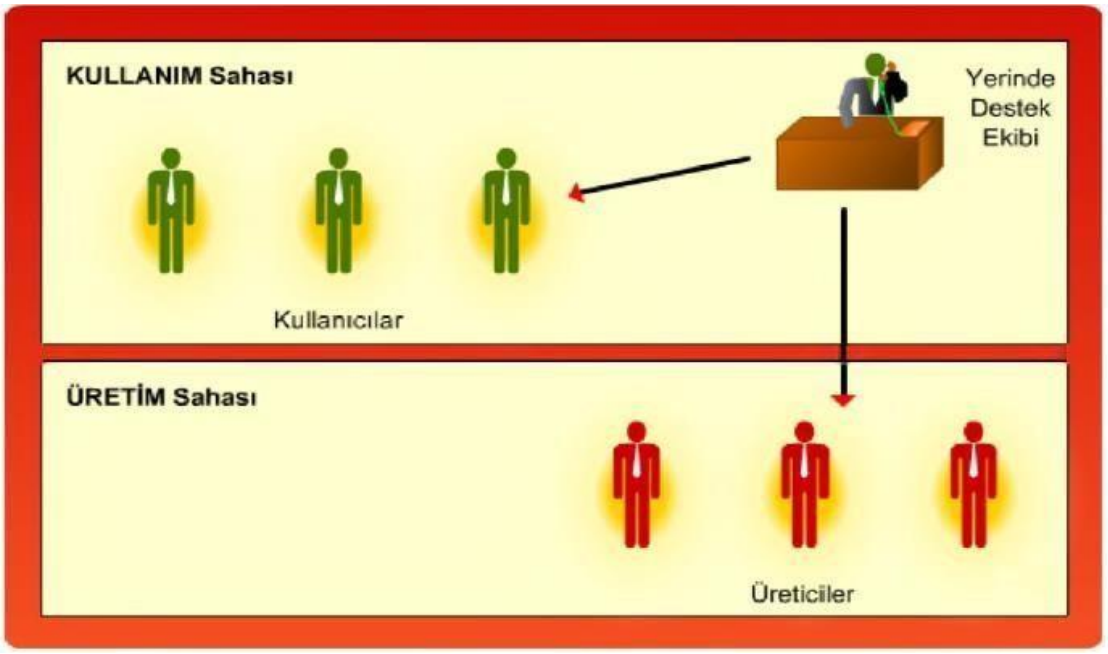
2.12 Test Planı

Proje test ekipleri ve görevleri şu şekildedir;

- Bir kişi kullanıcı ve admin olarak görev yapacak ve sistemi test edecektir.

2.13 Bakım Planı

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



Şekil 2.7 Üretim Sahası

3. SİSTEM ÇÖZÜMLEME

3.1 Mevcut Sistem İncelemesi

Mevcut sistem incelemesi öncelikle üniversitemizde olacaktır. Bunun yanı sıra benzer kaynaklardan incelenerek sistemin genel yapısı incelenmiş ve bizim sistemimize nasıl katkı yapılacağı araştırılmış sisteme uygulanmıştır.

3.1.1 Örgüt Yapısı

Örgüt yapısı olarak üniversitemiz tarafından oluşturulan bir örgüt yapısı vardır.

3.1.2 İşlevsel Model

Şekil 3.1 Üretim İşlevsel Model

3.1.3 Veri Modeli

Veri tabanı ilişkisel veri modelinde veriler tablolar üzerinden kurulan ilişkiye dayanmaktadır.

3.1.4 Varolan Yazılım/Donanım Kaynakları

- Ms Word
- Bilgisayar
- Visual Studio
- NetBeans

3.1.5 Varolan Sistemin Değerlendirilmesi

Sisteme kulanıcılar tarafından kolay kullanım gibi avantajlar sağlayarak verimli kullanım

alacaklardır.

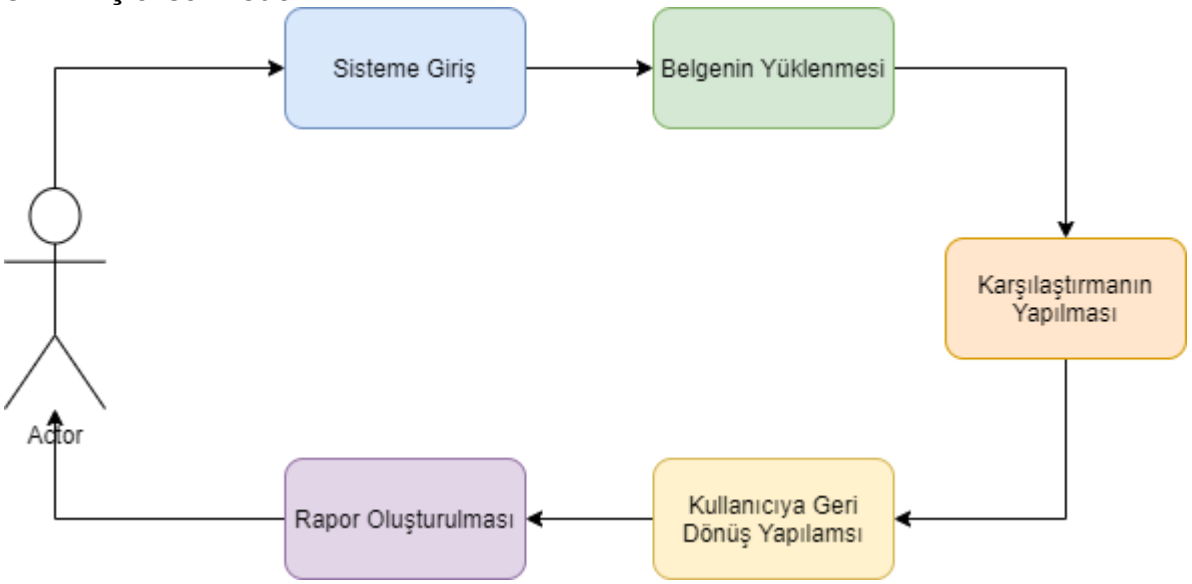
3.2 Gereksenen Sistemin Mantıksal Modeli

3.2.1 Giriş

Mevcut sistemler incelendiğinde sonuca giden yolda epeyce bir eksikler ve resmi olmayan durumlar söz konusu artık bu sistemie Türkiye hukuk standartlarına uydurmak bize kalıyor.

Sistemin işlevsel modeli ile başlamak gerekirse…

3.2.2 İşlevsel Model



Şekil 3.2 Üretim İşlevsel Model

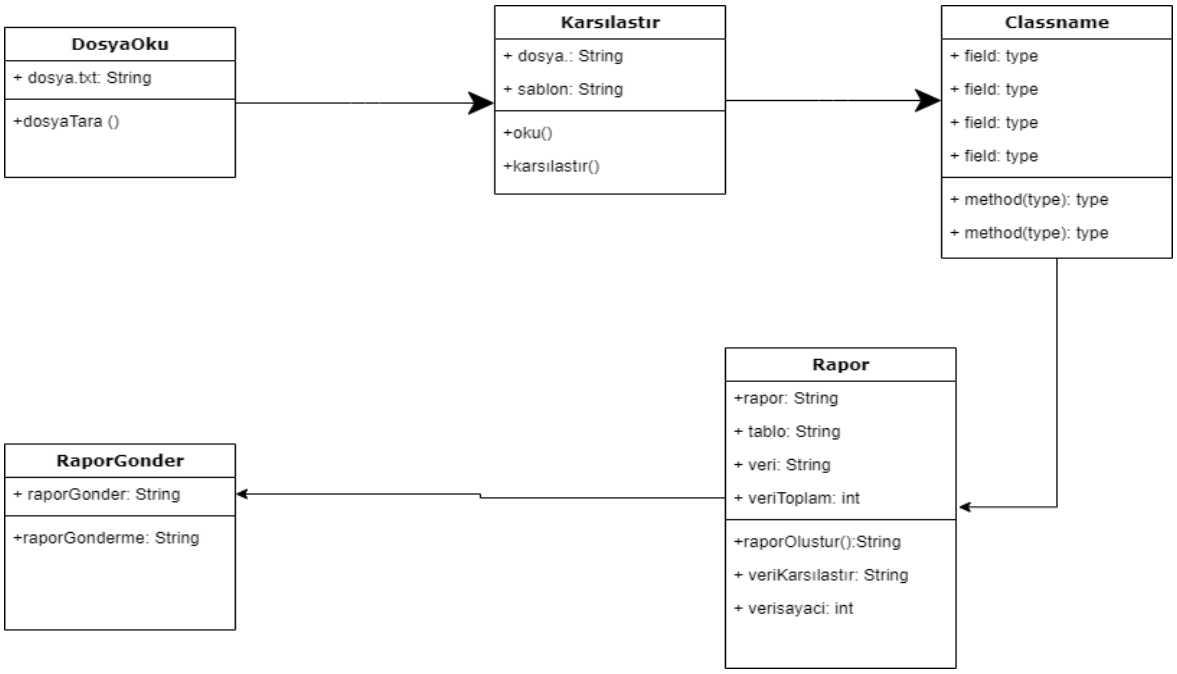
3.2.3 Genel Bakış

Genel hatlarıyla sistemi inceleyecek olursak mevcut sistemde mevcut olan tüm olaylar burada da var. Bunun yanı sıra olayın akış şekli USE-CASE diyagramında mevcuttur.

3.2.4 Bilgi Sistemleri\Nesneler

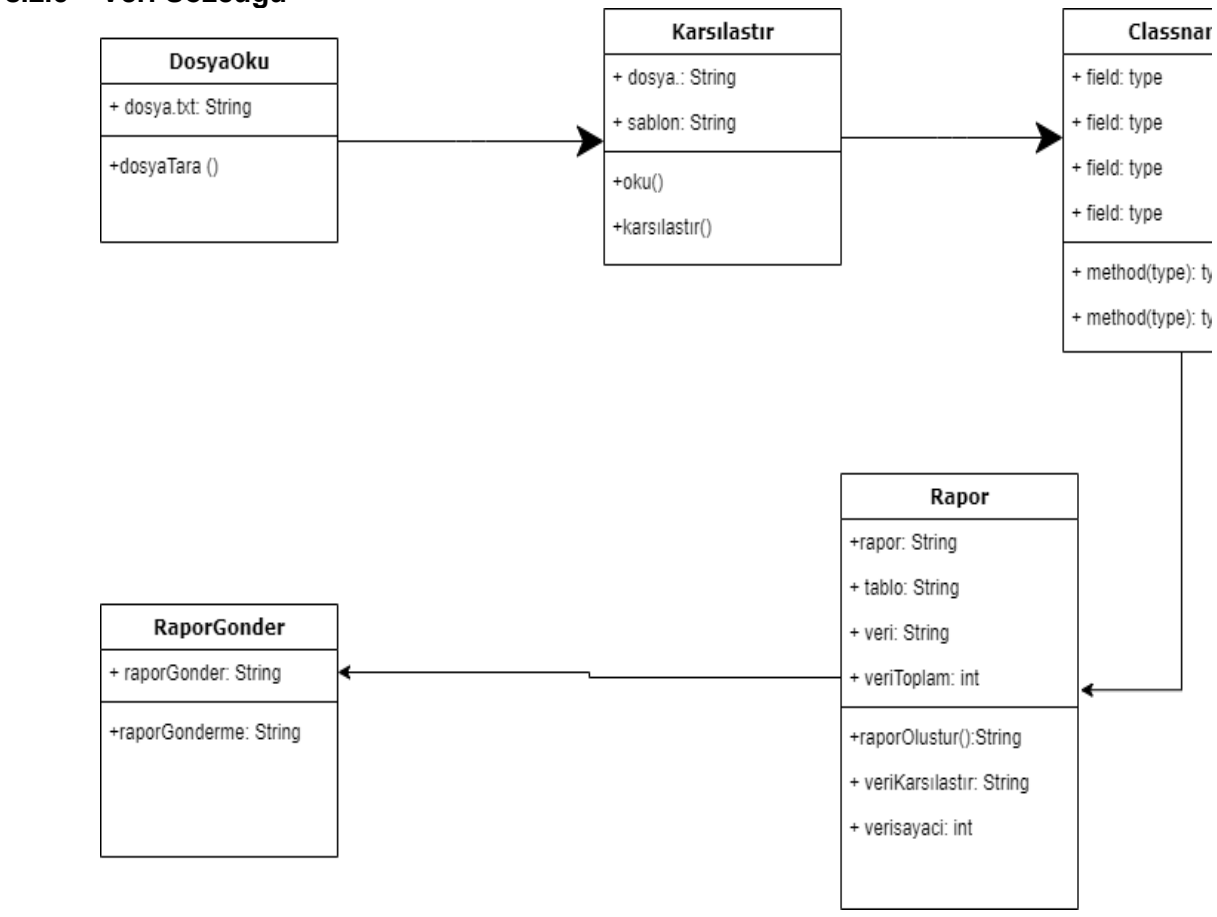
KULLANICI: Yapının asıl nesnesidir. Dosya karşılaştırma isteminde bulunur.

3.2.5 Veri Modeli



Şekil 3.3 Veri Modeli

3.2.6 Veri Sözcüğü



Şekil 3.4 Veri Sözcüğü Modeli

3.2.7 İşlevlerin Sıradüzeni,

Kullanıcı sisteme dosyayı yükler ve sırası ile işlemler gerçekleşir.

3.2.8 Başarım Gerekleri

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

- Sistemin sonuç üretim doğrulukları
- Tepki sürelerinin en aza indirilmesi
- Mali külfetin azaltılması
- Hile hata ve yanlışlıkların en aza indirilmesi
- Kullanım kolaylılığı
- Anlaşılabilirlik
- Tarafsızlık

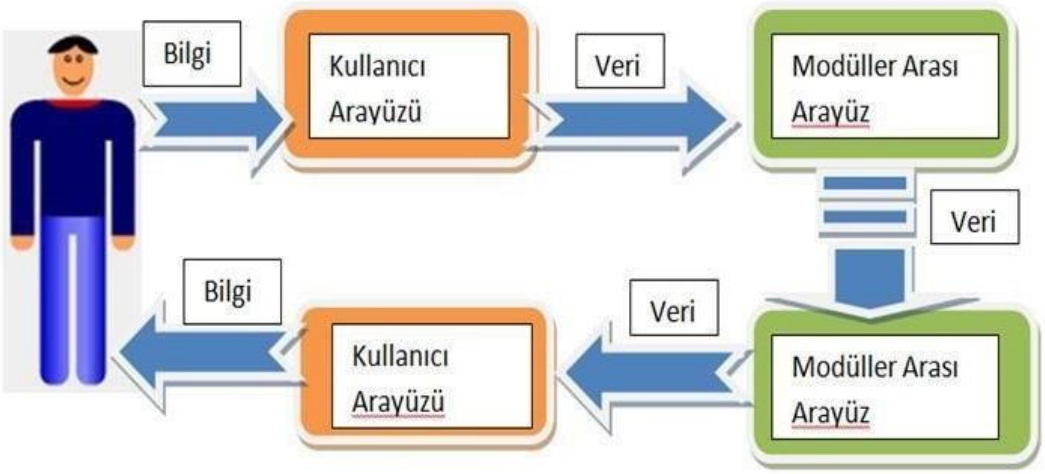
temel gereklilikler olarak tespit edilmiştir.

3.3 Arayüz Gerekleri

3.3.1 Yazılım Arayüzü

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

3.3.2 Kullanıcı Arayüzü



Şekil 3.5 Arayüz Şeması

Projede kullanıcının arayüzü tasarlanırken herhangi bir şekilde renkler seçilerek tarafsız rahat büyük puntolu yazılı bir arayüz tasarlanacaktır.

3.3.3 İletişim Arayüzü

Herhangi bir iletişim modülü oluşturulmadı**.**

3.3.4 Yönetim Arayüzü

Sistemin yönetim arayüzü yoktur.

3.4 Belgeleme Gerekleri

3.4.1 Geliştirme Sürecinin Belgelenmesi

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

3.4.2 Eğitim Belgeleri

Mevcut bir belgemiz bulunmamaktadır.

3.4.3 Kullanıcı El Kitapları

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

4. SİSTEM TASARIMI

4.1 Genel Tasarım Bilgileri

4.1.1 Genel Sistem Tanımı

1) **Product Backlog;** Proje için gerekli olan gereksinimler listesidir. Proje sonunda “Ne üretilmek isteniyor?” sorusuna cevap aranır. Product owner tarafından müşteriden gereksinimler alınır, öncelik sırasına göre sıralanır. Product owner, değişen ihtiyaçlara göre product backlog’a ekleme veya çıkarma yapabilir. Böylece değişim, projenin her aşamasında projeye kolayca entegre edilebilir olur.

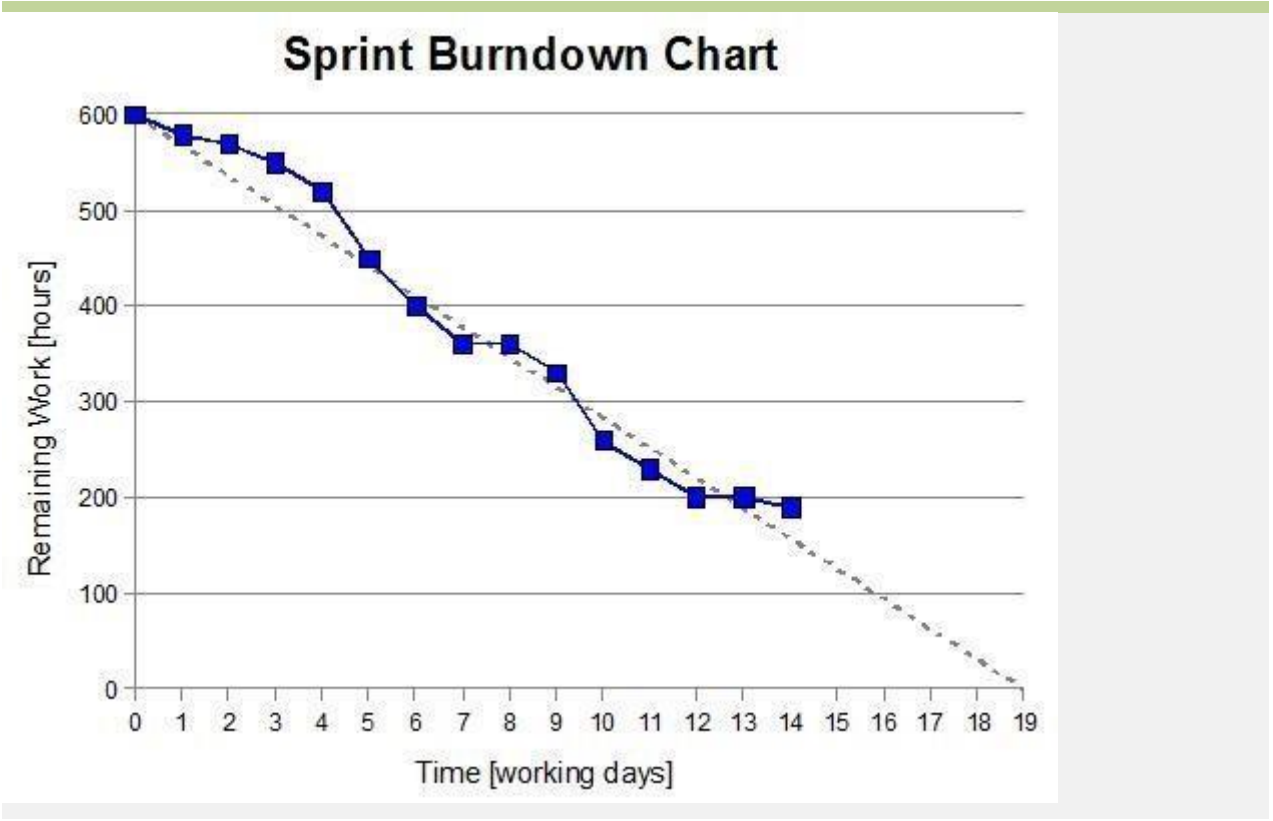
2) **Product Backlog Item;** Product backlog içindeki her bir gereksinime verilen isimdir.

3) **Sprint(Koşma);** Proje sprint denilen küçük kısımlara ayrılır. Scrum içerisindeki tüm aktiviteler sprint içerisinde gerçekleşir.1–2 haftalık süreçlerdir.

4) **Sprint Backlog;** Geliştirme takımı tarafından product backlog itemlar öncelik sırasına göre sprint içerisine alınırlar. Bir sprint boyunca yapılacak itemların listesini oluşturur. İşlerin detaylı olarak zaman çizelgesi çıkarılır.

5) **Scrum board;** Bir sprint içerisinde yapılacak olan maddeler burada yönetilir. Yapılacak olan tasklar “TO DO” bölümüne alınır. Takım üyesi bu işe başladığında “IN PROGRESS” bölümüne getirilir. Bir iş, test için hazırsa “TO VERIFY” durumuna getirilir. İş, kontrol edildikten sonra “DONE” bölümüne getirilir. Scrum toplantılarında bu maddeler durumlarına göre yerleri değiştirilir.

6) **Burndown Chart;** Yatay ekseninde sprintin günlerini, dikey ekseninde sprintte kalan işi gösteren grafiktir. Scrum’un temel ilkelerinden olan şeffaflığı sağlar.



Şekil 4.1 Scrum Çalışma Zaman Grafiği

- **Roller**

Pig Roller; Scrum sürecine dahil olanlar yani projede asıl işi yapan kişilerdir. Bunlar Scrum Master, Product Owner, Geliştirme Takımı'dır.

1) Product Owner; Geliştirme takımı ve müşteri arasındaki iletişimi sağlar. Projenin özelliklerini tanımlar. Projenin önceliklerine göre product backlogu oluşturur. Sprint'i iptal yetkisine sahiptir. Sprint neden iptal edilmek istenebilir? Hızla değişen ortamlarda bir sprinte alınan işlerin iş birimi için önemi kalmamış olabilir ya da sprinte alınan işlerden daha önemli işler ortaya çıkabilir. İş sahibi bunu görüp sprinti iptal etmek isteyebilir.

2) Scrum Master; Scrum kurallarını, teorilerini ve pratiklerini iyi bilir ve takımın bu kurallarını uygulamasından sorumlu kişidir. Takımın yöneticisi değildir. Takımı rahatsız eden, verimli çalışmalarını engelleyen durumları ortadan kaldırır.

3) Geliştirme Takımı; Bir Sprint'e alınan bütün işleri tamamlayacak özelliklere sahip kişilerdir. sprint backlogu oluştururlar. Kendi kendini yönetir. İşin verilmesini beklemezler, işi kendileri alır ve

geliştirirler. Kişilerin tek bir görevi yoktur, çapraz görev dağılımı yaparlar, herkes her şeyi yapabilir konumdadır. 5–7 kişi arasında değişir. Projenin geliştirilmesi ile ilgili sorumluluk geliştirme takımına aittir.

Chicken Roller; Scrum'ın işleyişinde aktif olarak yer almayan kişilerdir. Müşteriler, satıcılar gibi.

Şekil 4.2 Temsili Toplantı

- **Toplantılar**

1) Sprint Planning; Product backlog ile belirtilen gereksinimler, bu toplantı ile geliştirme takımı tarafından küçük görevlere (task) ayrılır. Takımdaki her bir kişi kendi hızına göre bu taskleri kendilerine alır. Bu toplantıya product owner, geliştirme takımı ve scrum master katılır. Sprintler; her sprint sonunda product owner a sunulmak üzere yazılım geliştirmeyi hedefleyecek şekilde belirlenir.1–3 haftalık sprintler oluşturulur.

2) Daily Scrum; Her gün aynı yerde aynı saatte ayak üstü yapılan 15 dakikalık toplantılardır. Üyeler davet edilmeyi beklemezler. Bu toplantı gelecek 24 saati planlamak üzere yapılır. Takımdaki her üye dün ne yaptım,bugün ne yapacağım, işimi engelleyen herhangi bir sorun var mı sorularına cevap verir. Bu sayede herhangi bir sorunu var ise scrum master bu problemi ortadan kaldırır. Takım üyelerinden bu probleme yardımcı olabilecek biri var ise toplantı sonunda iletişime geçebilirler. Daily scrum her ne koşulda olursa olsun yapılır. Takımdaki birinin geç kalması veya gelmemesi toplantıyı etkilemez. Sadece takımdaki büyük çoğunluk yok ise toplantı yapılmaz.

3)Sprint Review; Her sprint sonunda yapılır. Yapılan sprint gözden geçirilir, ortaya çıkan ürün değerlendirilir. Amaç yazılımın ürün sahibinin gereksinimlerine uygun olarak geliştirildiğinden emin

olmaktır. Eğer bir hata var ise farkedilir ve düzeltilir.

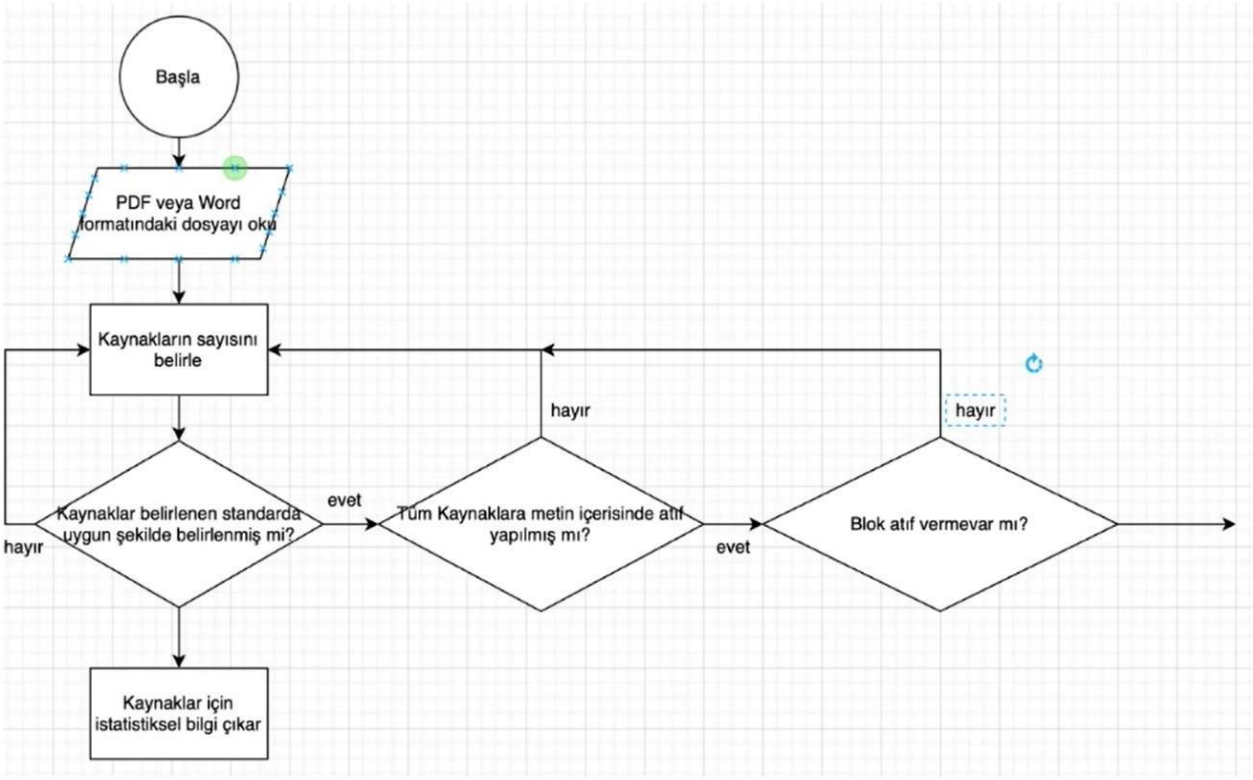
4)Sprint Retrospective; Sprint boyunca yapılan işlerin kalitesinin, doğruların ve yanlışların değerlendirildiği toplantıdır. Bu toplantı scrum takımının kendini geliştirebilmesi için bir fırsattır. “Neleri daha iyi yapabiliriz?”, “Nasıl daha iyi yapabiliriz?” sorularına cevap aranır. Bu aşamadan sonra bir sonraki sprint planning toplantısı gerçekleştirilerek yazdıklarımızın hepsi tekrardan yaşanır.

4.1.2 Varsayımlar ve Kısıtlamalar

Sistemde varsayılan değerler bulunmamaktadır.

4.1.3 Sistem Mimarisi

Sistemin mimarisinin akış diyagramı şeklinde verilmesinin temel nedeni sistemin işleyiş mantığının nasıl olduğu ve nasıl bir yol çizileceğinin bilinmesidir.Akış diyagramı sistemin temel mantığı hakkında bize fikir verecektir.



Şekil 4.3 Akış Şeması

4.1.4 Dış Arabirimler

1.4.1 Kullanıcı Arabirimleri

Kullanıcı arabirimin ilk başında sistem giriş ekranı bulunacak. Ve her birimin kendine ait ekranları olacaktır.

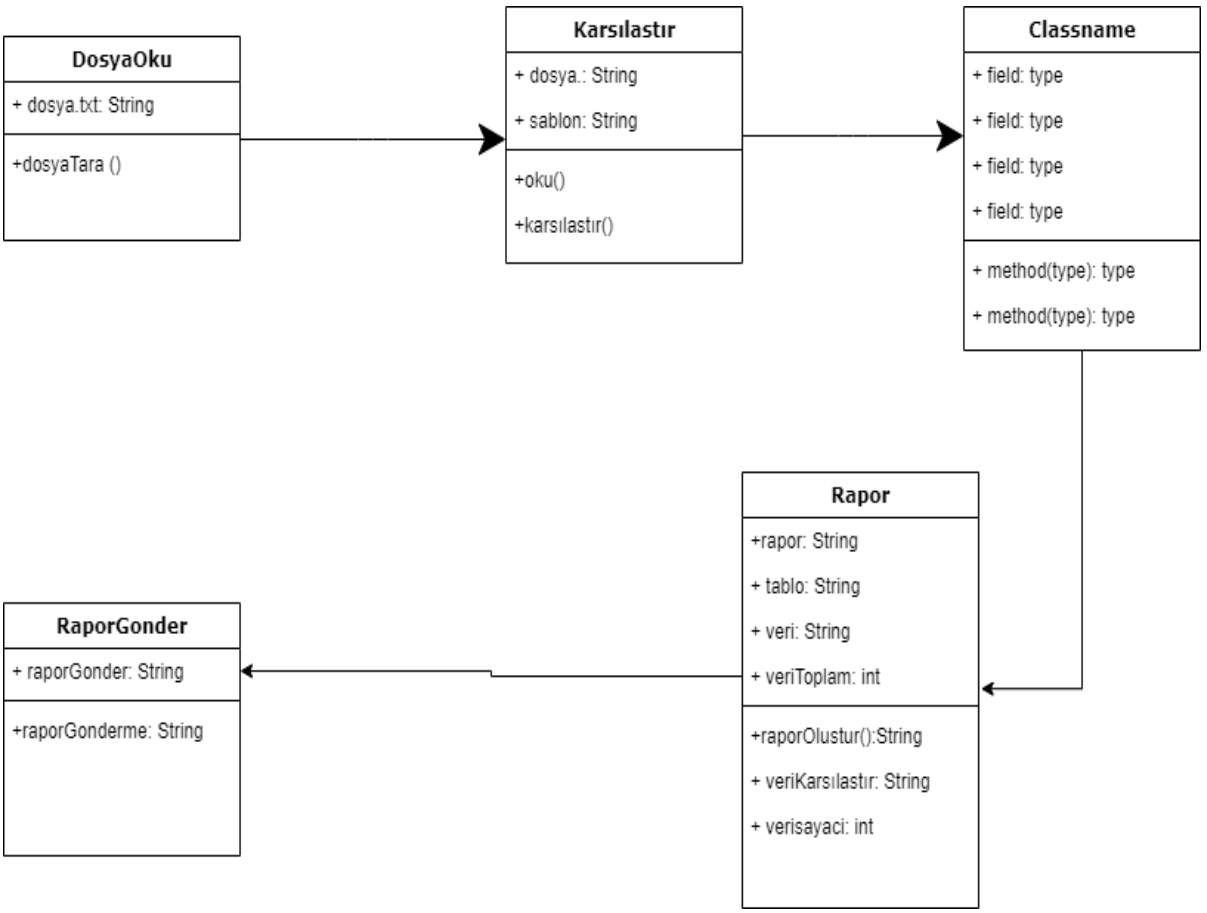
1.4.2 Veri Arabirimleri

Veri arabirimlerinde sistem kullanıcı isteği üzerine kayıt sağlayabilecektir. Bununla birlikte veriler MySql very tabanında tutulabilecektir.

1.4.3 Diğer Sistemlerle Arabirimler

Şuan tam bilgi sahibi olmadığımız için bu arabirim kullanılmayacaktır.

4.1.5 Veri Modeli



Şekil 4.4 Veri Modeli

4.1.5 Testler

Genel hatlarıyla testlerimiz iki aşamada gerçekleştirilecek. Bilinen adıyla pilot bölge uygulaması yapılacak.

Alfa Aşaması: Sistemin geliştirildiği yerde kullanıcıların gelerek katkıda bulunması sistemi test etmesi ile yapılacak.

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

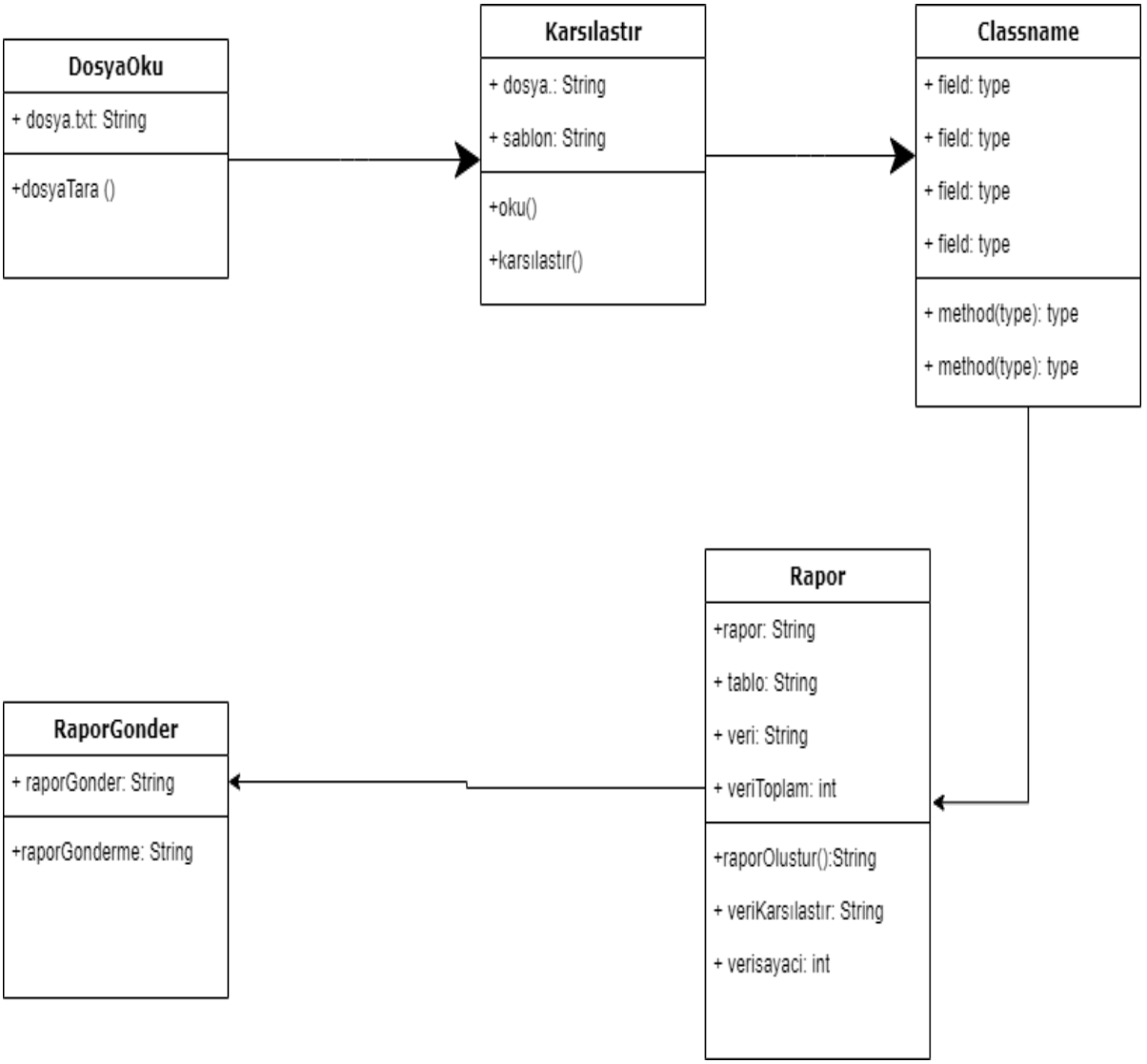
4.1.6 Performans

Sistemin performansını etkileyen veriler değerlendirilerek;

- Stabilitesi ve işleyiş performansı
- Çalışma zamanındaki uyumluluk soruna değerlendirilecektir.

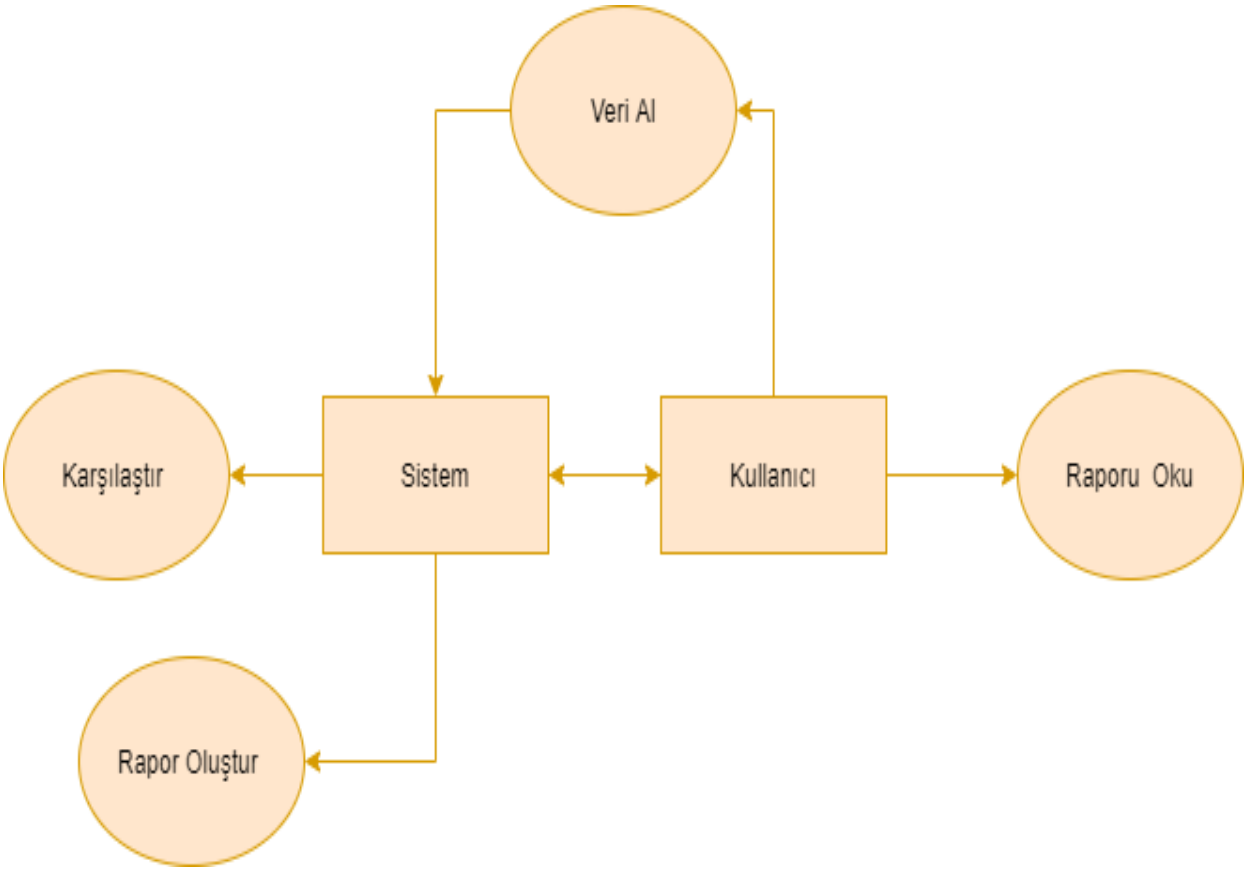
4.2 Veri Tasarımı

4.2.1 Veri Tanımları



Şekil 4.5 Veri Tanımları

4.2.2 Tablo-İlşki Şemaları



Şekil 4.6 Tablo-İlişki Şeması

4.2.1 Veri Tanımları

Veri tipi olarak String kullanılmıştır.. String olarak kullanılan değerler kelime içeren değerleri tutacağından ötürü kullanıldı. Boolean veri tipi işte işin can alıcı noktası burası boolean veri tipi iki faklı değer alır ya truedir ya false eğer ki görevlendirme varsa true olur yada karşılaştırıldı ise true değeri dönecektir.

4.2.2 Değer Kümesi Tanımları

Sayac: Kullanılan veriler doğrultusunda oluşacak olan toplam değerdir.

4.3 Sürec Tasarımı

4.3.1 Genel Tasarım

Genel olarak tasarımda ilk önce veri tabanı modeli oluşturuldu. Ardından giriş modülü en sonda kullanıcı arayüzü oluşturduk.

4.3.2 Modüller

4.3.2.1 Giriş Modülü

4.3.2.1.1 İşlev

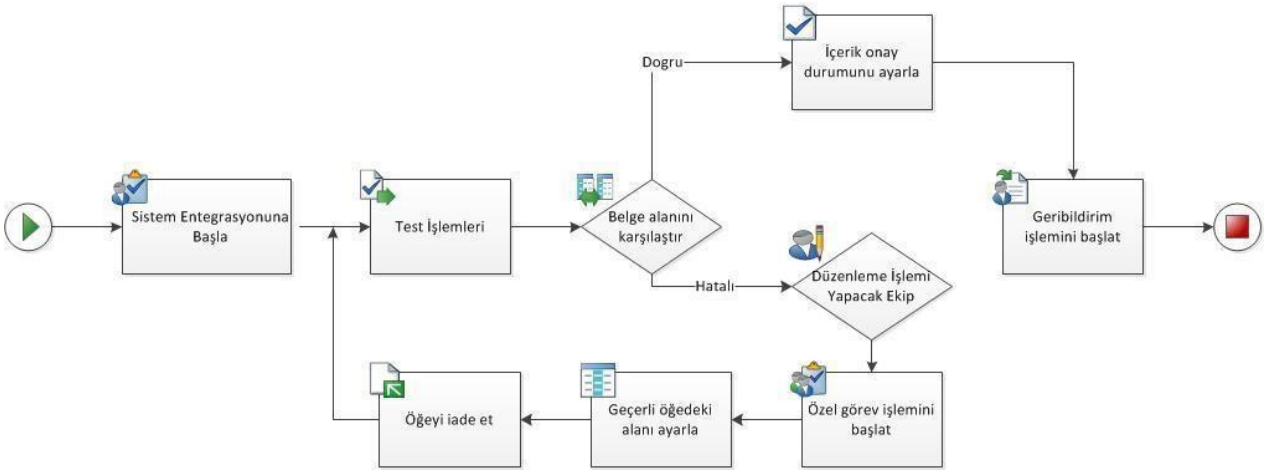
Kullanıcının sisteme müdahale edecebileceği ekrana erişmesi için aşması gereken bir modüldür.

4.3.3 Kullanıcı Profilleri

KULLANICI: Yapının asıl işlem yapan nesnesidir.

4.3.4 Entegrasyon ve Test Gereksinimleri

Sistemimizin daha öncede belirttiğimiz gibi devletin mevcut veri tabanlarıyla entegrasyon halinde olması gerekir bu yüzden devlet yetkililerinden bir ekibe gereksinimimiz var. Bunun yanı sıra pilot bölgedeki test süresince 1.derece yazılım yetkilileri orda bulunmalı ve sistem testini gözlemleyip notlar çıkartmalı.



Şekil 4.7 Test Şeması

4.2 Ortak Alt Sistemlerin Tasarımı

4.2.1 Ortak Alt Sistemler

Kullanılacak tek ortak sistem devletin mevcut sistemidir bu sistemle tc numarası bazında kişinin tüm bilgisi alınacaktır.

4.2.2 Modüller arası Ortak Veriler

Modüller arasında ilişkili veri modelinde olduğu üzere ortak veriler mevcuttur. Bunlar Tc_no, Görevlendirme, Görev ve sonuç birbirleriyle ilişkili ortak verilerdir.

4.2.3 Ortak Veriler İçin Veri Giriş ve Raporlama Modülleri

Ortak veriler için böyle bir modül kullanılmadı

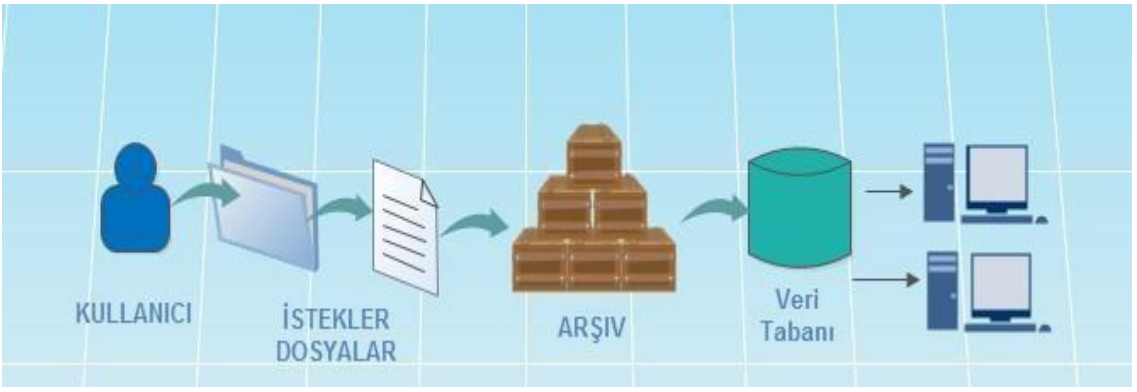
4.2.4 Güvenlik Altsistemi

Yazılım sistemlerinin güvenilirliğe ilişkin nicelikleri, kullanıcıların gereksinimlerini karşılayacak şekilde ortaya koymak ve güvenilirliğin hesaplanmasına yönelik verileri toplama, istatistiksel tahminleme, ölçütlerin tespiti, yazılıma ait mimari özelliklerin belirlenmesi, tasarım, geliştirme ve bunlara yönelik çalışma ortamının belirlenmesi ve modellenmesini kapsamaktadır.

1. Model seçme ve düzenlemeye yönelik faaliyetlerin temelinde uygun hedeflerin tespit edilmesi bulunmaktadır.
2. Hata ve aksaklıkların analiz edilmesi için uygun verilerin tanımlanması gerekmektedir. Örneğin, arıza veya hataları önemine göre sınıflandırmak, hatalar arası ortalama süreyi bulmak, hata nedenlerini araştırmak, hataları bulmaya yönelik test verilerine karar vermek.
3. Belirtilen hedeflere yönelik veriler modellenir.
4. Geçmişe yönelik verilerin zaman bilgilerini de içerecek şekilde elde edilerek yazılım geliştirme sürecine dâhil etmek.
5. Yazılım geliştirme sürecinin modellenmesi, hata ile karşılaşılıp, test sürecine başlamak ve model doğrulama işlemlerine gerçekleştirmek.
6. Güvenilirlik tahminleme modelinin seçilmesini sağlamak.
7. Güvenilirlik modeli tarafından kullanılacak olan parametrelerini tespit etmek.
8. Verilen bir noktayı kullanarak gelecekteki olası hatalar hakkında tahmin yapmak.
9. Tahmin edilen hata ve arıza oranları ile gerçekleşen değerleri kıyaslamak.

4.2.5 Veri Dağıtım Altsistemi

Veri dağıtımından ziyade veri alma üzerine bir sistem kuruldu. Dağıtılan sistemlerde zararlı kişilerin müdahalesi kolay olmaktadır bunun yanı sıra bizim sistemimizde terminaller serverlardan veri çekecek serverlar terminallere veri göndermeyecek.



Şekil 4.8 Veri Dağıtım Alt Sistemi

4.2.6 Yedekleme ve Arşivleme İşlemleri

Depolanan verilerin, herhangi bir nedenle zarar görmesi, sistemin çalışma süreçlerinde ciddi zararlar oluşturabilir. Yaşanabilecek bir felaket durumu sonrasında, depolanan verilerin geri yüklenememesi, sistemin sağlandığı kullanıcılara veya kurumlara çok ciddi zararlar verebilir.

Bu nedenle sistemin çalışma süreçlerine bağlı olarak, yedekleme sistemleri kurulmalı ve yedekleme işlemleri günlük olarak takip edilmelidir.

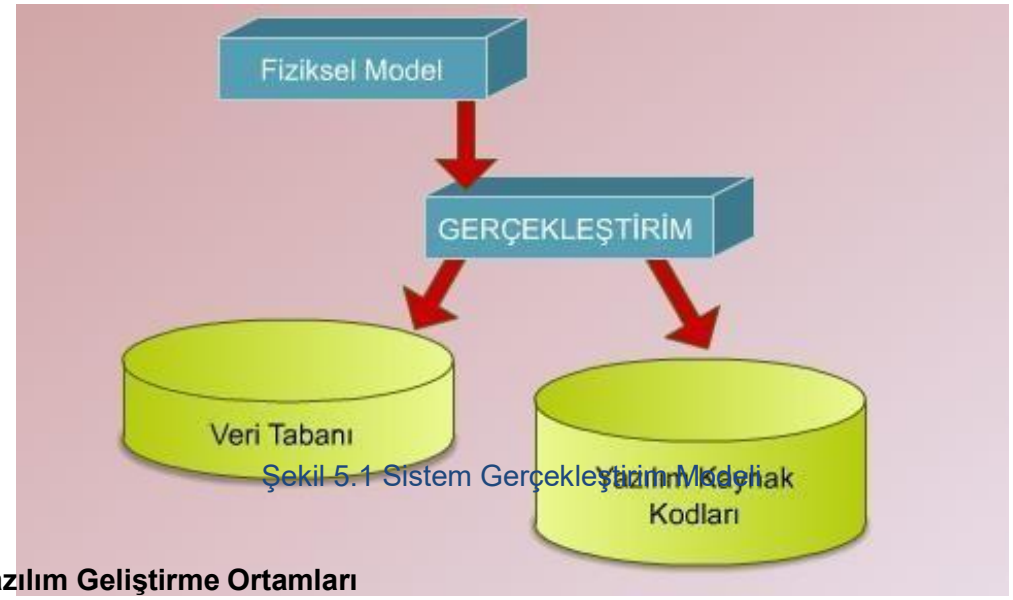
Yedekleme sistemlerinin kurulumu; yedeklenecek veri miktarı, yedekleme sıklığı, yedeklenen verinin zaman içerisinde değişme oranı ve maksimum veri kaybı gibi parametrelere bağlıdır.

Sistemin birden fazla sunucusunun eş zamanlı yedekleme işlemini yapabilmesi, işletim sistemlerinin kayıt dosyalarını tam ve eş zamanlı olarak yedekleyebilmesi ve işletim sistemleri üzerinde çalışan veri tabanı uygulamasının yedeklerini sistem kapatılmadan alabilmesi gerekmektedir.

5. SİSTEM GERÇEKLEŞTİRİMİ

5.1 Giriş

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



Şekil 5.1 Sistem Gerçekleştirim Modeli

4.1 Yazılım Geliştirme Ortamları

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

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

CASE Araçları belirlendi ve yazılım geliştirme ortamı hazırlandı.

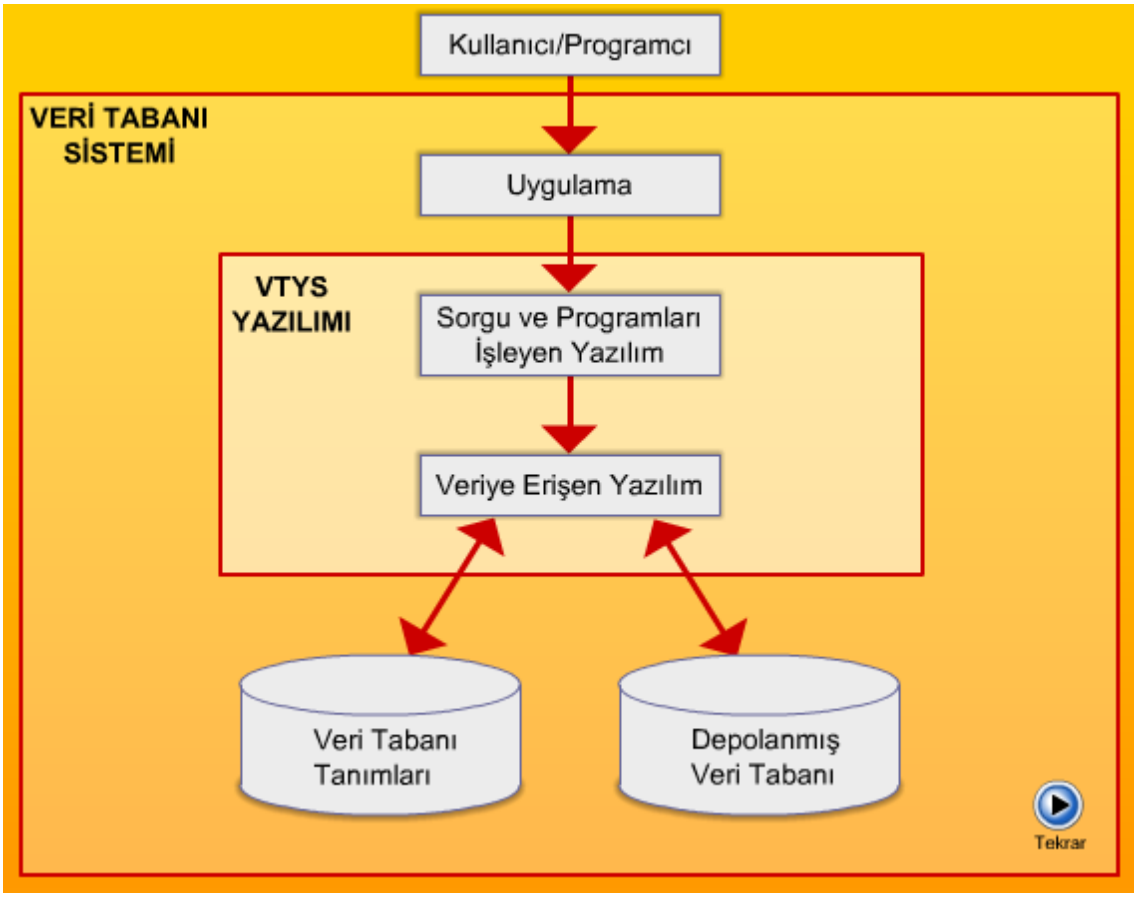
4.1.1 Programlama Dilleri

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

4.1.2 Veri Tabanı Yönetim Sistemleri

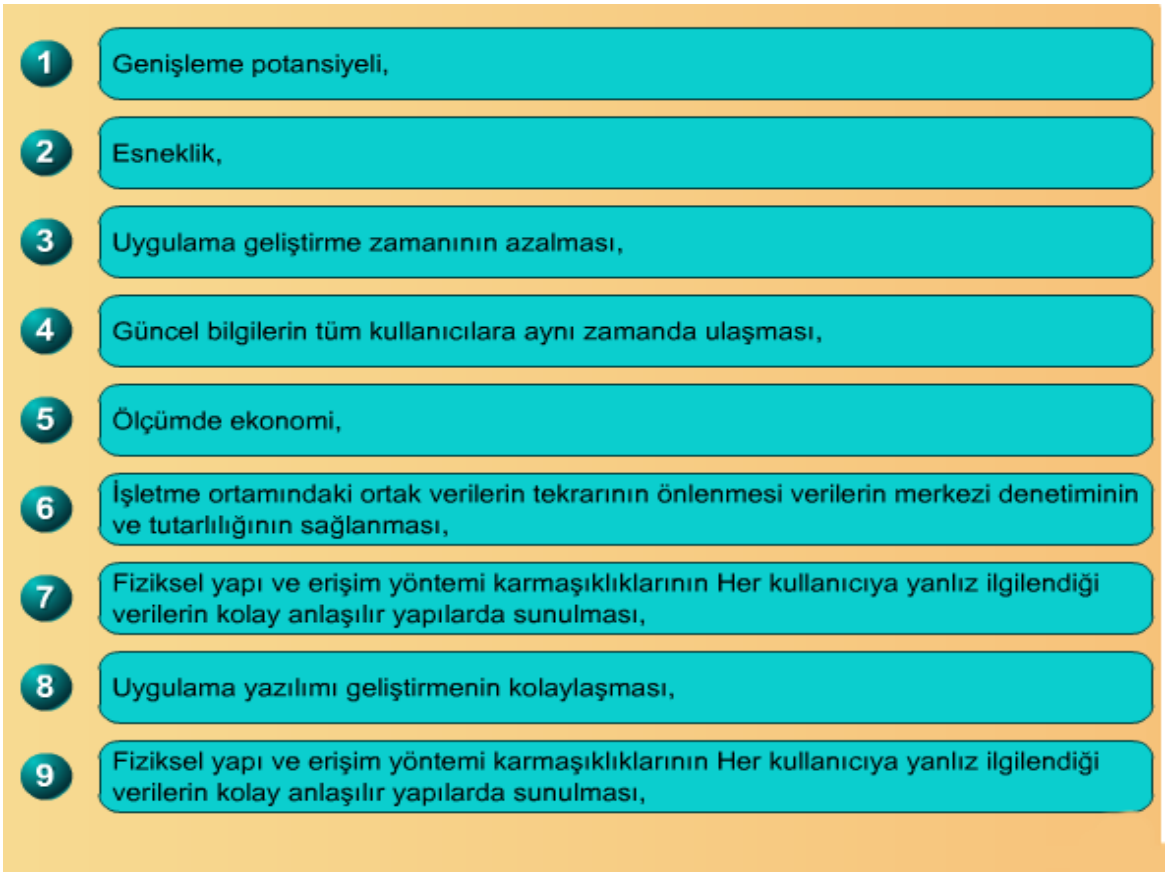
Veri tabanı tanımları ve Depolanmış veri tabanı ise SQL ile çözümleniyor.

Veri tabanı yönetiminde kullandığımız hiyerarşiyi bir göstermek gerekirse:



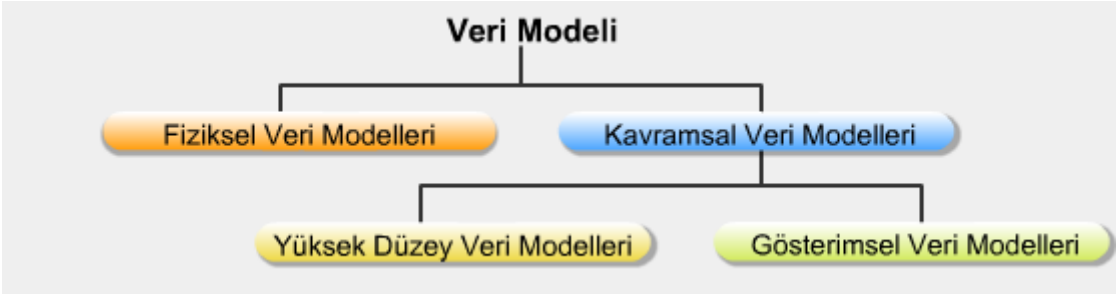
Şekil 5.2 Veri Tabanı Sistemi

4.1.2.1 VTYS Kullanımının Ek Yararları



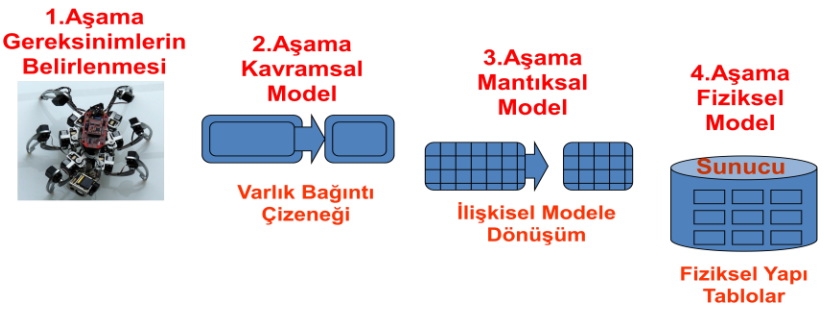
Şekil 5.3 VTYS nin Ek Yararları

4.1.2.2 Veri Modelleri



Şekil 5.4 Veri Modeli

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



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

1. Gereksinimlerin belirlenmesi
- Veri tipleri
- Veri grupları
- Veriler ile ilgili kurallar
- Veriler üzerinde yapılması gereken işlemler

2. Kavramsal Model

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

3. Mantıksal Model

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

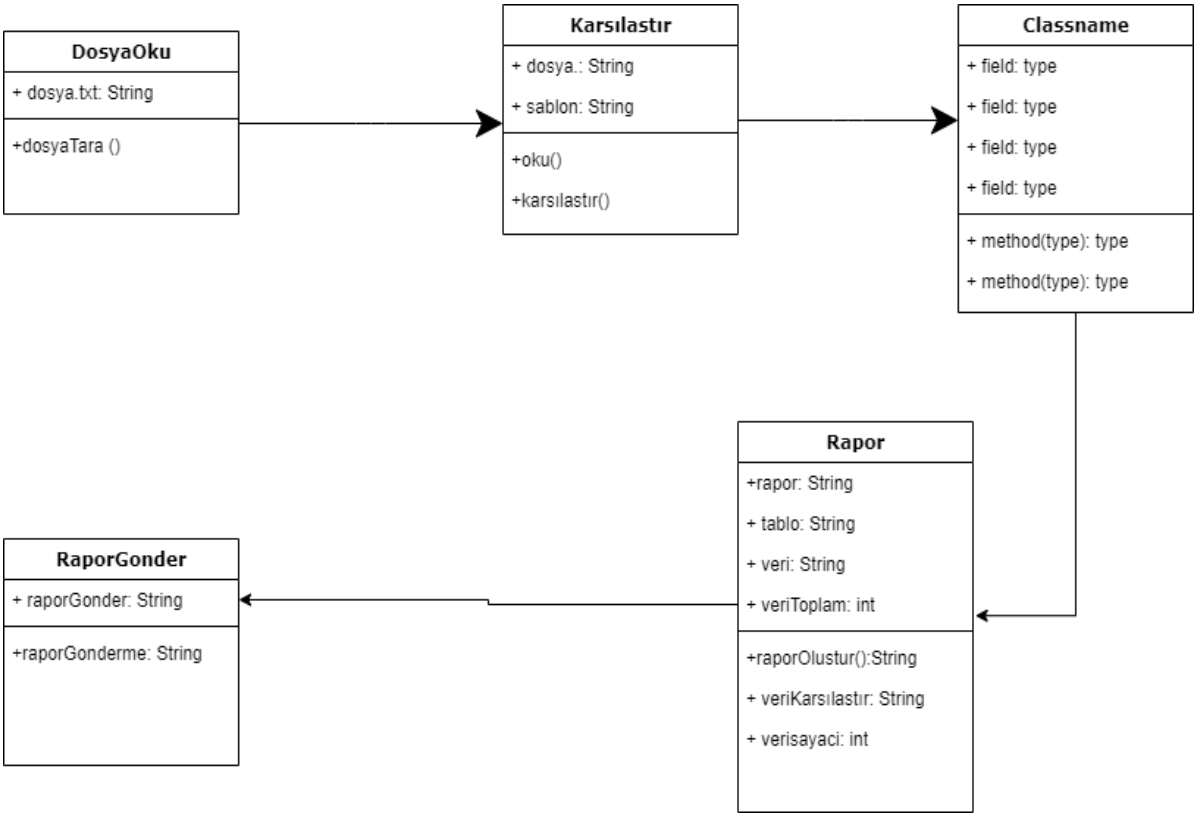
4. Fiziksel Model

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

«uses» ullanıcı «uses»

Admin Use

4.1.2.3 Şemalar



Şekil 5.7 Şemalar

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

4.1.2.4 Veritabanı Dilleri ve Arabirimleri

Sistemimizde veritabanı dili olarak SQL kullanıldı. Henüz prototip aşamasında olan sistemimiz için MYSQL arabirimini yeterli gördük.

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.

1.1.1.1 Veri Tabanı Sistem Ortam

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

1.1.1.2 VTYS'nin Sınıflandırılması

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

1.1.1.3 Hazır Program Kütüphane Dosyaları

Zaten entegre olduğu için böyle bir şeye ihtiyaç duymadık.

1.1.1.4 CASE Araç ve Ortamları

Case araçları olarak ise Microsoft un Visio Project ürünlerini kullandık.

1.2 Kodlama Stili

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

5.3.1 Açıklama Satırları

Açıklama satırları karmaşık her satırın sonunda yapıldı. Ve biten her günün sonunda kodun kalındığı yere o günün tarihi atıldı.

5.3.2 Kod Biçimlemesi

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

5.3.3 Anlamlı İsimlendirme

Sistem kodlamasının genel yapısında kullanılan değişkenlerin veri tabanında karşılığı varsa önce “tabloadı_islevadı_sayısı” seklinde bir anlamlı isimlendirme yaptık.

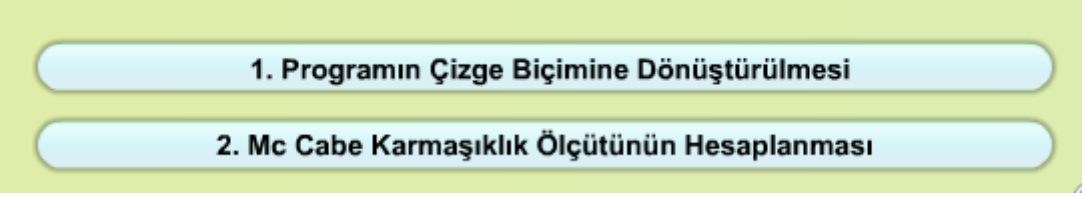
5.3.4 Yapısal Programlama Yapıları

Genel olarak 3 başlıkta incelersek:

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

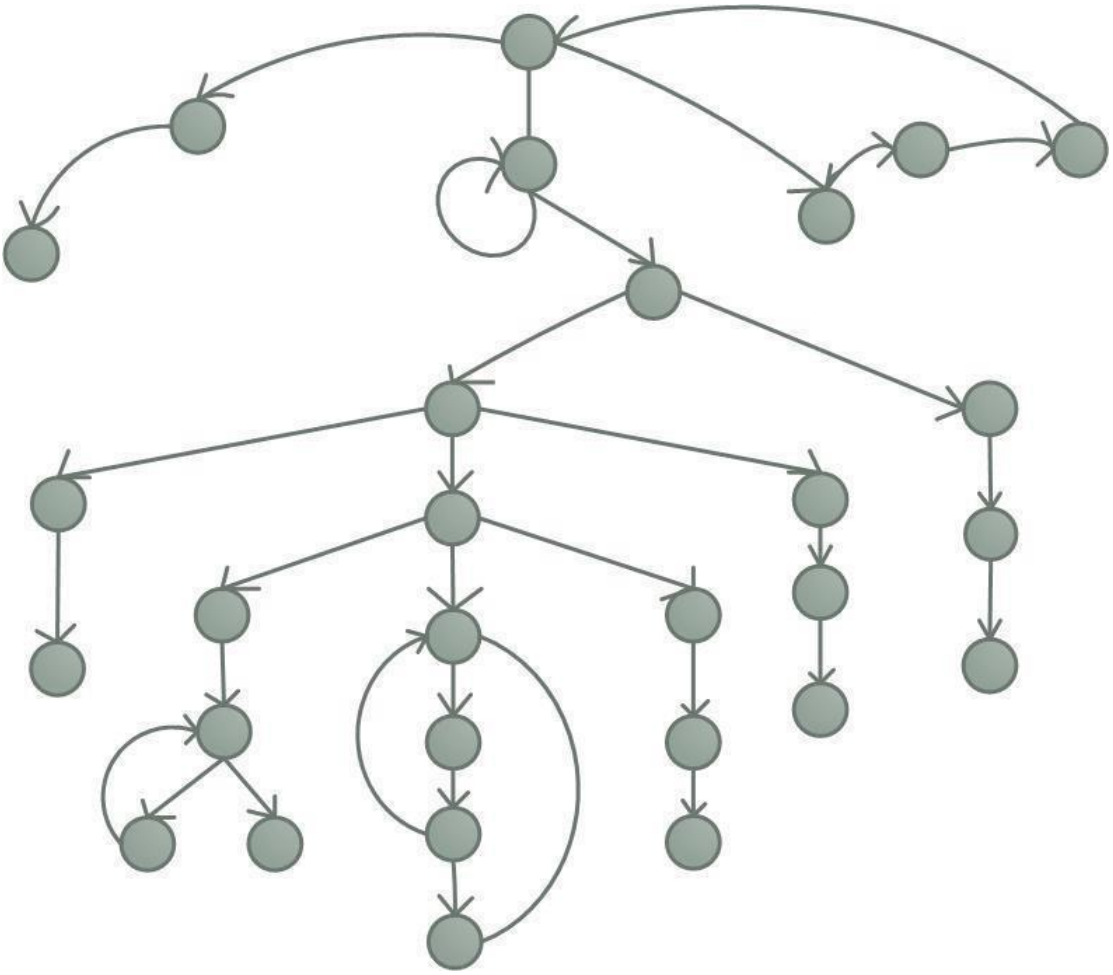
1.3 Program Karmaşıklığı

Program karmaşıklığını ölçmek için bir çok teorik model geliştirilmiştir. Bu modellerin en eskisi ve yol göstericisi McCabe karmaşıklık ölçütüdür. Bu bölümde bu ölçüt anlatılmaktadır. Söz konusu ölçüt 1976 yılında McCabe tarafından geliştirilmiştir. Bu konuda geliştirilen diğer ölçütlerin çoğu, bu ölçütten esinlenmiştir.
McCabe ölçütü, bir programda kullanılan "koşul" deyimlerinin program karmaşıklığını etkileyen en önemli unsur olduğu esasına dayanır ve iki aşamada uygulanır:



Şekil 5.9 Program Karmaşıklığı

5.4.1 Programın Çizge Biçimine Dönüştürülmesi



Şekil 5.10 Programın Çizgi Hali

5.4.2 McCabe Karmaşıklık Ölçütü Hesaplama

k = 13 Kenar sayısı

d = 9 Düğüm sayısı

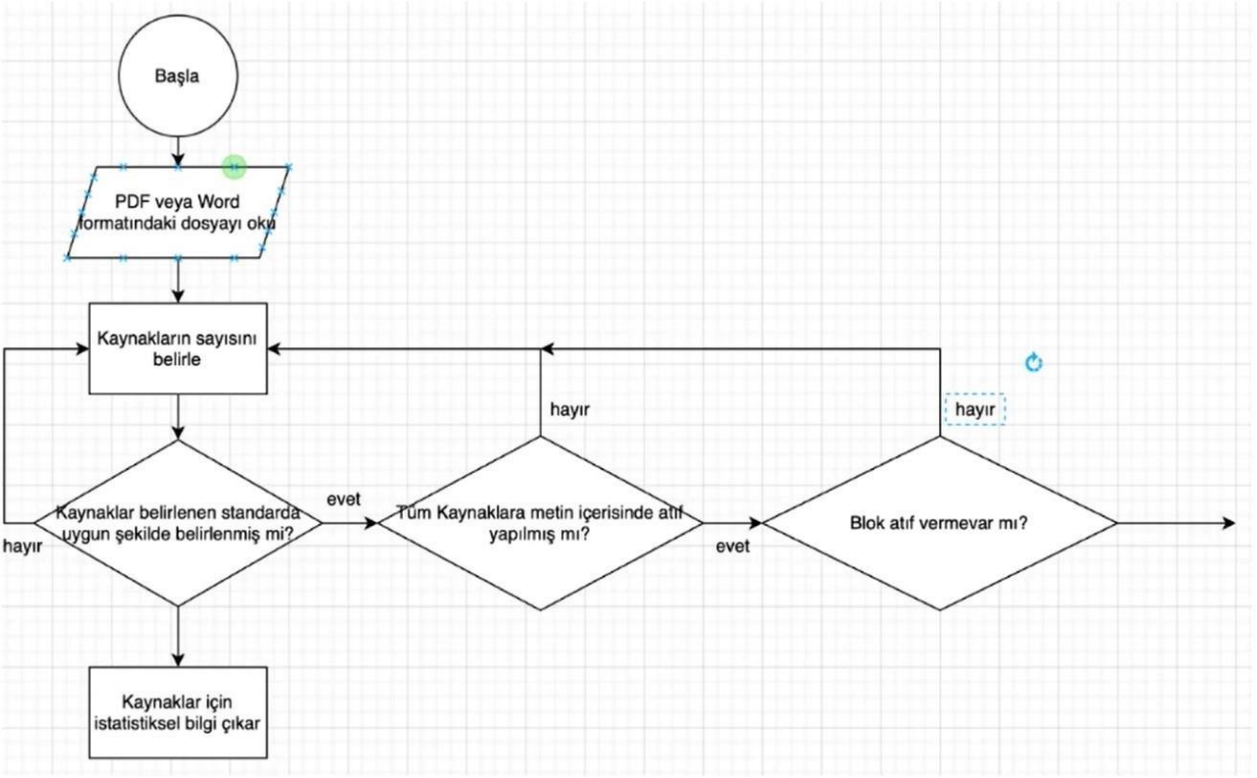
p = 1 Bileşen sayısı

V(G)=k-d+2p (Formülüyle bulunur)

V(G)= k – d + 2p

VG= 30 – 25+2*31

= 67



Şekil 5.11 Akış Diyagramı

5.5 Olağan Dışı Durum Çözümleme

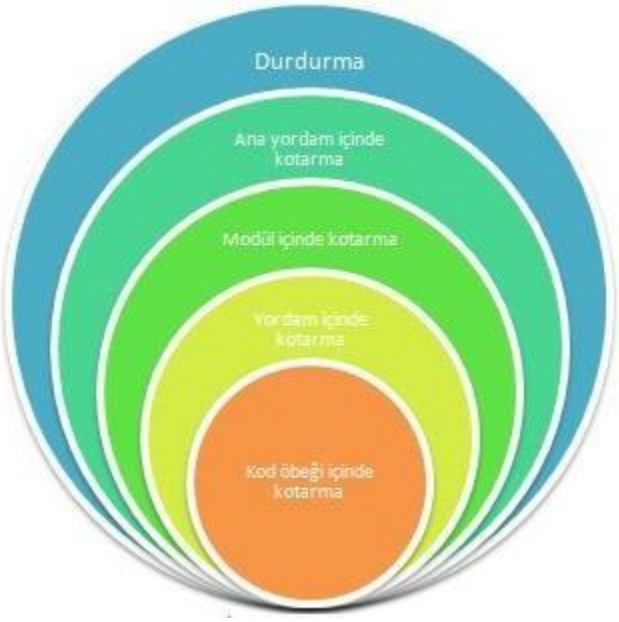
Olağan dışı durum, bir programın çalışmasının, geçersiz ya da yanlış veri oluşumu ya da başka nedenlerle istenmeyen bir biçimde sonlanmasına neden olan durum olarak tanımlanmaktadır.

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 5.12 Olağan Dışı Halde Yapılacaklar

1.5 Kod Gözden Geçirme

Hiç kimse, önceki sürümlerini gözden geçirmeden ve incelemeden okunabilir bir program yazamaz. Hiçbir yazı editörün onayını almadan basılamayacağı gibi hiçbir program da incelenmeden, gözden geçirilmeden işletime alınmamalıdır. Kod gözden geçirme ile program sınama işlemlerini birbirinden ayırmak gerekir.
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. biçiminde özetlenebilir. 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) aşağıdaki soruların yanıtları aranır. Bu sorulara ek sorular eklenebilir. Bazı soruların yanıtlarının "hayır" olması programın reddedileceği anlamına gelmemelidir.

5.6.2.1 Öbek Arayüzü

Oluşturduğumuz öbekleri test etmek için belli sorular sorduk bu sorular:

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

Şeklinde oldu.

5.6.2.2 Giriş Açıklamaları

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

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

Şeklinde oldu.

Veri Kullanımı

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

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

Şeklinde oldu.

5.6.2.3 Öbeğin Düzenlenişi

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

Gözden geçirme sırasında referans alınacak sorular olacaktır.

5.6.2.4 Sunuş

Artık son kısma gelindiğinde ise şu sorular soruldu:

- 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 m

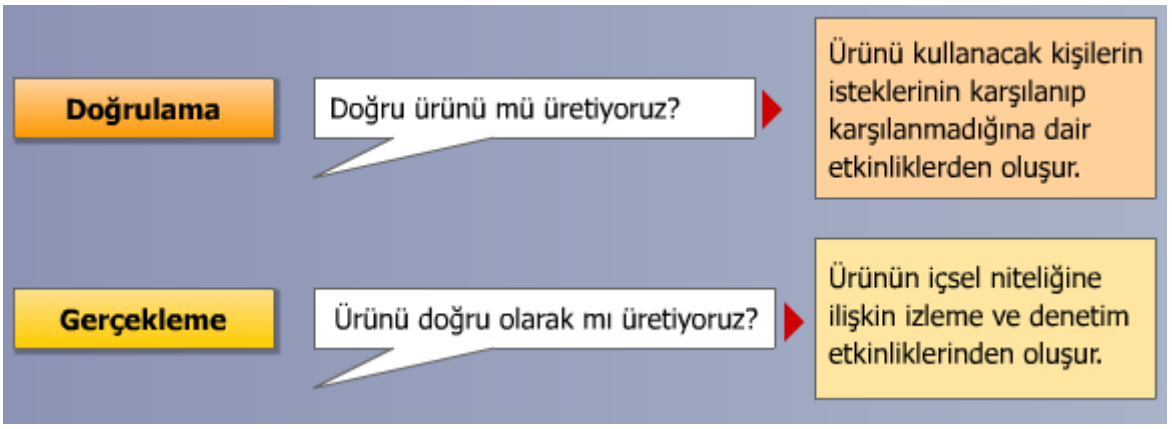
6. DOĞRULAMA VE GEÇERLEME

6.1 Giriş

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

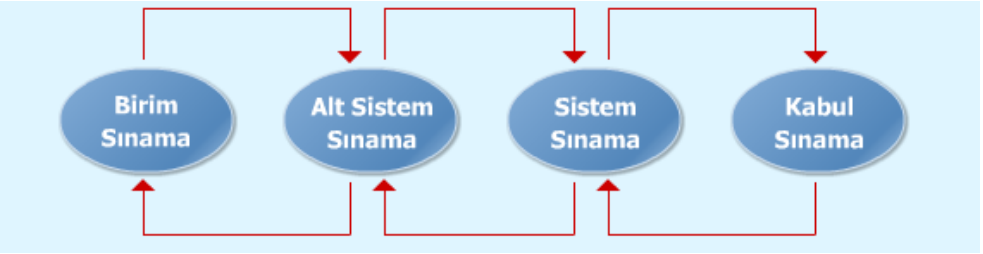
- Yazılım belirtimlerinin ve proje yaşam sürecindeki her bir etkinlik sonunda alınan çıktıların,
- tamam, doğru, açık ve önceki belirtimleri tutarlı olarak betimler durumda olduğunun doğrulanması.
- Proje süresince her bir etkinlik ürününün teknik yeterliliğinin değerlendirilmesi ve uygun çözüm
- elde 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 Doğrulama Geçerleme

6.2. Sınama Kavramları



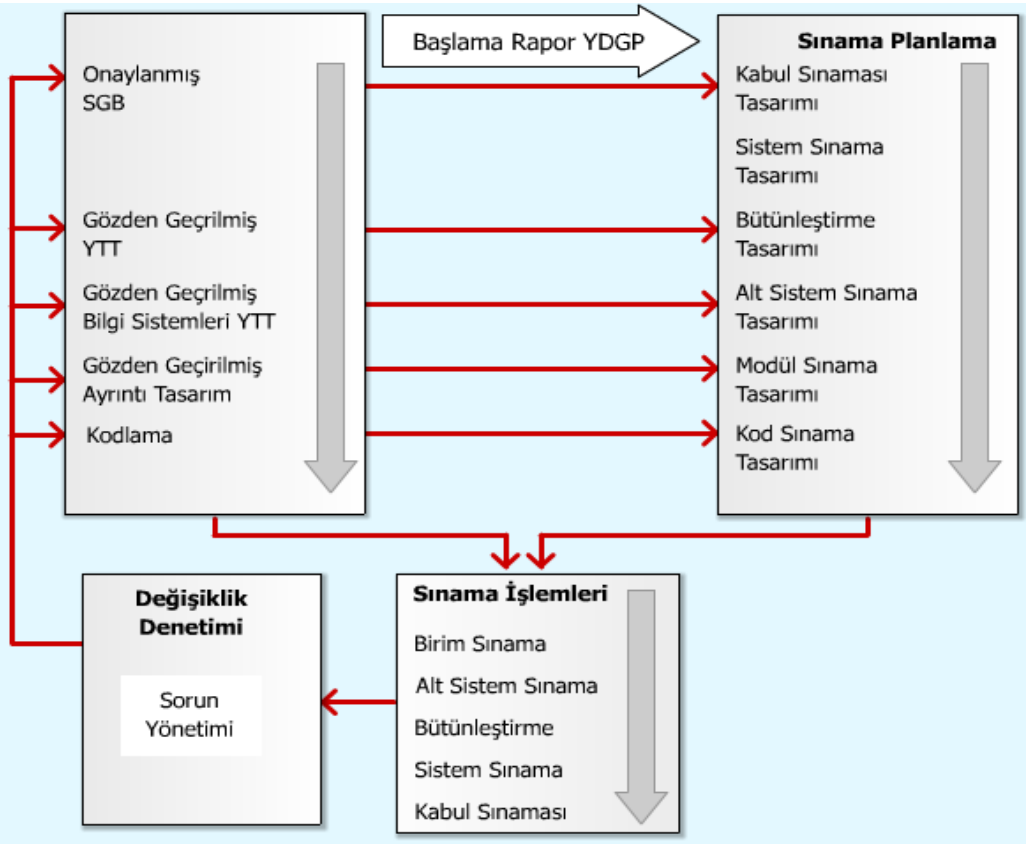
Birim Sınama: Sistemin birimleri olan YSK-YSK il-YSK İlçe-Sandık Kurulu-Seçmen sırasıyla kendi içlerinde birimleri sınandı ve sonuçları çıkartıldı.

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 6.2 Yaşam Döngüsü

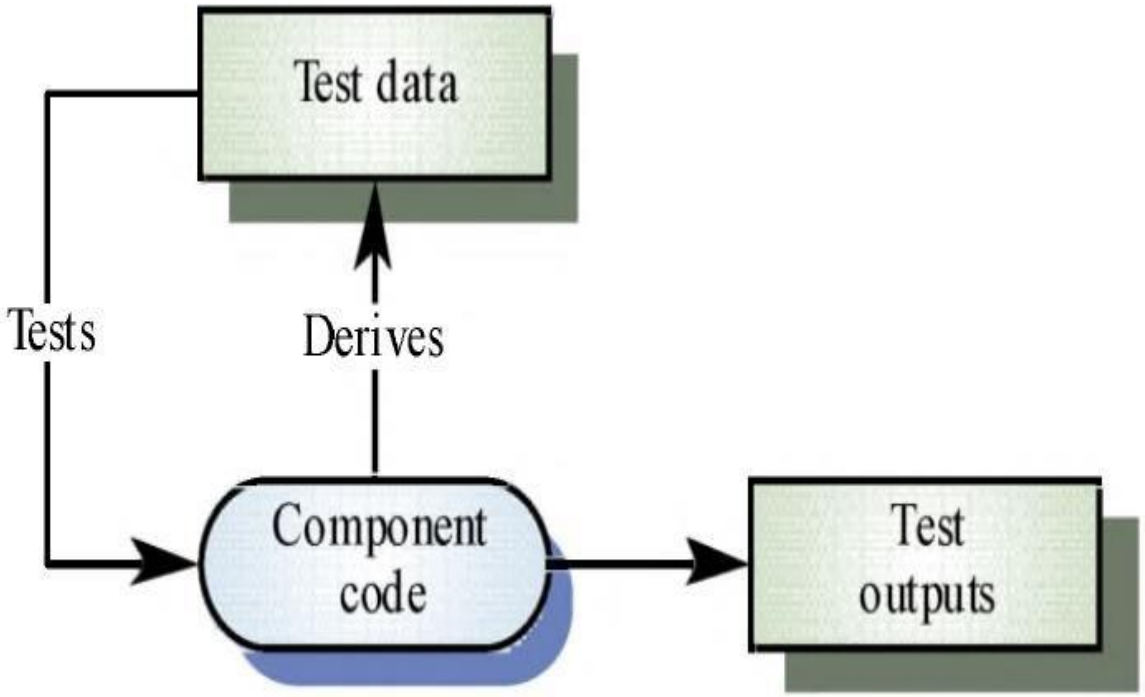
6.4. Sınama Yöntemleri

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.

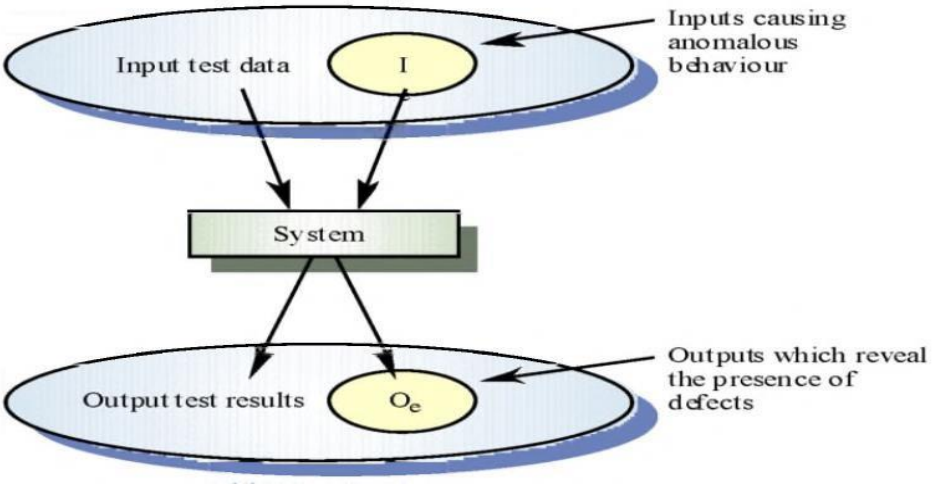


Şekil 6.3 Beyaz Kutu Sınaması

6.4.2 Temel Yollar Sınaması

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

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



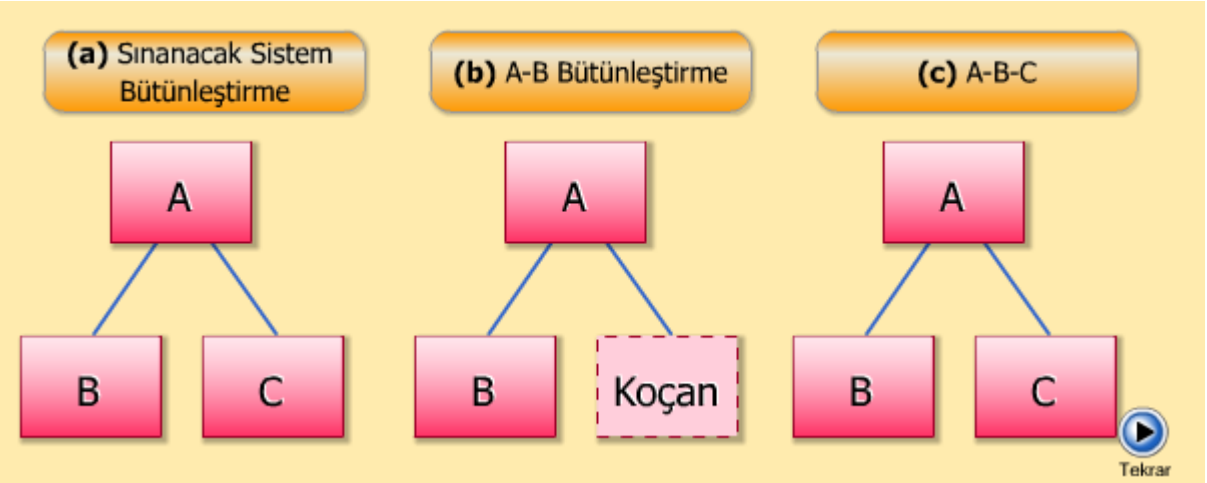
Şekil 6.5 Sınama Şekli

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

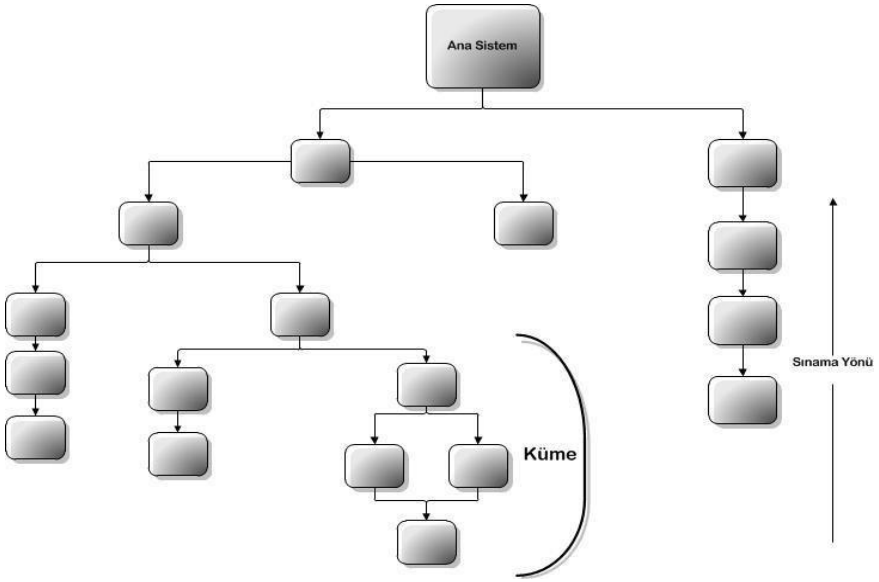
Yukarıdan aşağı bütünleştirmede, önce sistemin en üst düzeylerinin sınanması ve sonra aşağıya doğru olan düzeyleri, ilgili modüllerin takılarak sınanmaları söz konusudur. En üst noktadaki bileşen, bir birim/modül/alt sistem olarak sınandıktan sonra alt düzeye geçilmelidir. Ancak bu en üstteki bileşenin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



Şekil 6.6 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 6.7 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 önlemleri açıklar.

Giriş

Amaç
Tanım ve Kısaltmalar
Referanslar

Sınama Yönetimi

Sınama Konusu
Sınama Etkinlikleri ve Zamanlama

Temel Sınama Etkinlikleri
Destek Etkinlikler

Kaynaklar ve Sorumluluklar
Personel ve Eğitim Gereksemeleri
Sınama Yaklaşımı
Riskler ve Çözümler
Onaylar

Sınama Ayrıntıları

Sınanacak Sistemler
Girdiler ve Çıktılar
Sınamaya Başlanma Koşulları

Girdilerin Hazır Olması
Ortam Koşulları
Kaynak Koşulları

Şekil 6.8 Sınama Planlaması

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 oluşturmas

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

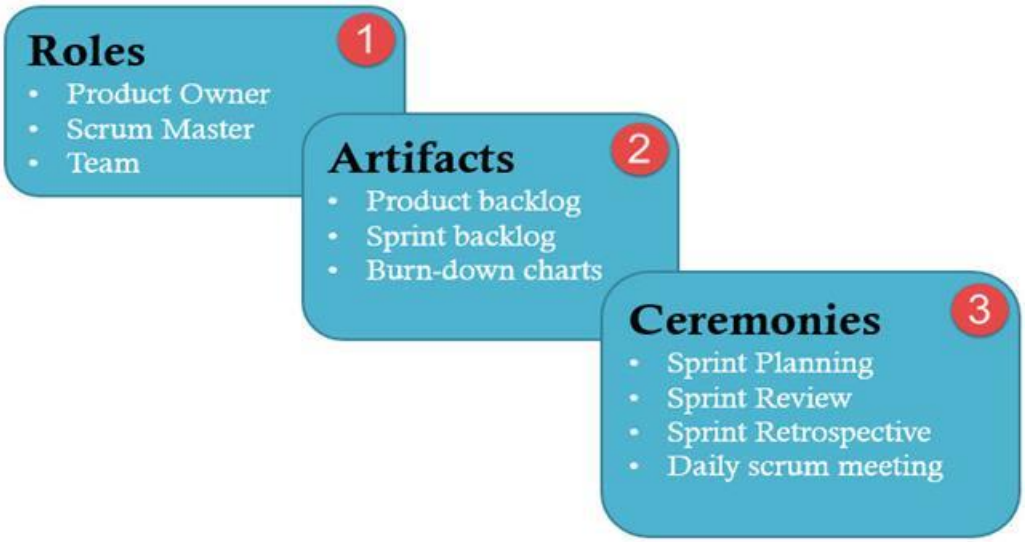
Scrum Testi , yazılım uygulama gereksinimlerinin karşılandığını doğrulamak için scrum metodolojisinde yapılan bir testtir. Güvenlik, kullanılabilirlik, performans vb. Gibi işlevsel olmayan parametrelerin kontrol edilmesini içerir. İşlemde test edenin aktif bir rolü yoktur, bu nedenle genellikle geliştiriciler tarafından Unit Test ile gerçekleştirilir. Bazen projenin doğasına ve karmaşıklığına bağlı olarak özel test ekiplerine ihtiyaç duyulur.

Scrum Metodolojisinin Temel Özellikleri

Scrum'ın Temel Özellikleri şunlardır:

- Scrum, hızla değişen geliştirme ihtiyaçlarını karşılamak için **sprintler** olarak bilinen ayarlanabilir kapsamı olan kısa bir sabit yayın döngüsü programına sahiptir . Her sürümün birden fazla sprinti olabilir. Her Scrum Projesinin birden fazla Yayın Döngüsü olabilir.
- Yinelenen bir dizi **toplantı, etkinlik ve kilometre taşları**
- Her sprintten sonra bazı işlerin hazır olduğundan emin olmak için **hikayeler** olarak bilinen yeni gereksinimleri test etme ve uygulama uygulaması

Scrum aşağıdaki 3 Sütuna dayanmaktadır:



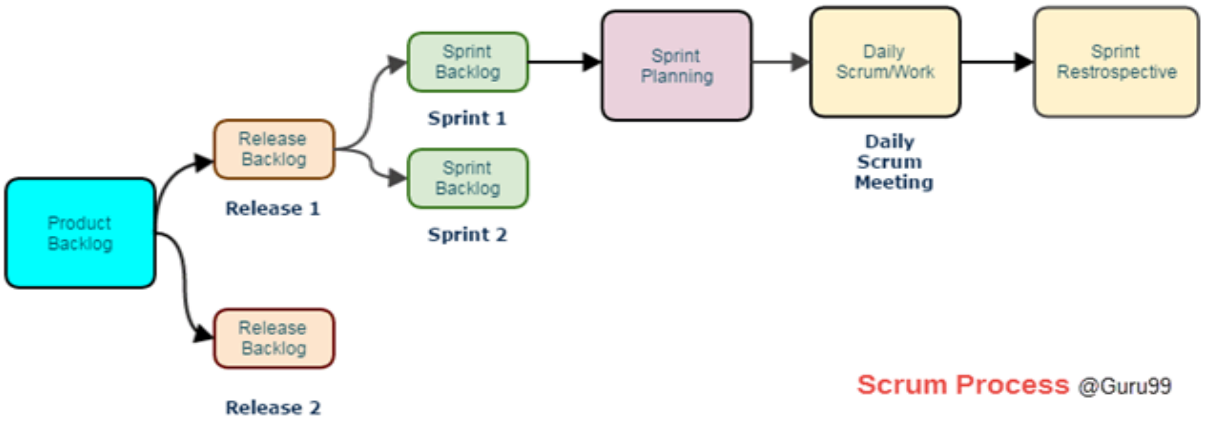
Şekil 6.7 Scrum Test Rolleri

1. Scrum'daki Roller

Scrum Testinde üç ana rol vardır - Ürün Sahibi, Scrum Master ve Geliştirme Ekibi. Onları detaylı olarak inceleyelim

Ürün sahibi	Saldırı ustası	Takım
• Ürünün özelliklerini tanımlar.	• Ekibi yönetir ve ekibin üretkenliğine bakar	• Ekip genellikle 5-9 üyedir
• Ürün Sahibi, yayınlanma tarihine ve ilgili özelliklere karar verir	• Blok listesini tutar ve gelişimdeki engelleri kaldırır.	• Geliştiricileri, tasarımcıları ve bazen testçileri vb. İçerir.
• Ürünün piyasa değeri ve karlılığına göre özellikleri önceliklendirirler.	• Tüm rol ve işlevlerle koordine eder	• Ekip çalışmalarını kendi başına organize eder ve planlar
• Ürünün karlılığından sorumludur.	• Ekibi dış müdahalelerden korur	• Sprint hedefine ulaşmak için proje sınırları içinde her şeyi yapma hakkı vardır
• İş öğesi sonucunu kabul edebilir veya reddedebilir	• Günlük scrum, sprint incelemesi ve planlama toplantılarına davet	• Günlük törenlere aktif olarak katılın

2. Scrum Eserleri



Şekil 6.7 Scrum Test Prosedürü

Bir saldırı süreci şunları içerir:

- **Kullanıcı hikayeleri:** Test edilen sistemin işlevlerinin kısa bir açıklamasıdır. Sigorta Sağlayıcı için örnek - "Prim, çevrimiçi sistem kullanılarak ödenebilir."
- **Ürün İş Listesi:** Bir scrum ürünü için yakalanan kullanıcı hikayelerinin bir koleksiyonudur. **Ürün sahibi** , ürün birikimini **hazırlar** ve sürdürür. Ürün sahibi tarafından önceliklendirilir ve ürün sahibinin onayı ile herkes ürüne ekleme yapabilir.
- **Release Backlog: Sürüm** , yineleme sayısının tamamlandığı bir zaman çerçevesidir. **Ürün sahibi** , bir sürüm için hangi hikayelerin hedeflenmesi gerektiğine karar vermek için scrum master ile **koordineli** çalışır. Sürüm biriktirme listesindeki hikayelerin bir sürümde tamamlanması hedeflenir.
- **Sprintler:** Ürün sahibi ve geliştirici ekibi tarafından karar verilen, genellikle 2-4 haftalık kullanıcı hikayelerini tamamlamak için belirlenen bir süredir.
- **Sprint İş Listesi:** Bir sprintte tamamlanacak bir dizi kullanıcı hikayesidir. Sprint iş yığını sırasında, iş asla atanmaz ve takım kendi başına işe kaydolur. Ekibe aittir ve ekip tarafından yönetilirken, kalan tahmini çalışma günlük olarak güncellenir. Sprint'te gerçekleştirilmesi gereken görevlerin listesidir.
- **Blok Listesi:** Scrum master tarafından sahip olunan ve günlük olarak güncellenen blokların ve yapılmamış kararların bir listesidir.
- **Burndown grafiği:** Burn-down grafiği, devam etmekte olan çalışmanın ve süreç boyunca tamamlanan çalışmanın genel ilerlemesini gösterir. Tamamlanmamış hikayeleri ve özellikleri bir grafik biçiminde temsil eder

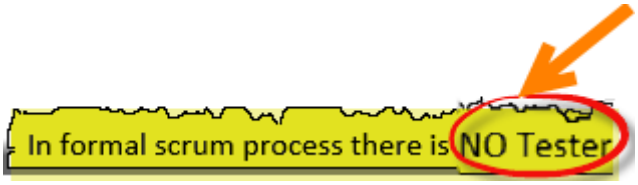
3. Scrum'da Törenler (Süreçler)

- **Sprint Planlama:** Bir sprint, takımın hikayeleri sürüm birikiminden sprint iş yığınına aktarmasıyla başlar; scrum master tarafından barındırılmaktadır. Test Uzmanları, Sprint İş Listesindeki çeşitli hikayeleri test etme çabasını tahmin ediyor.
- **Günlük Scrum: Scrum** master tarafından barındırılır, yaklaşık 15 dakika sürer. Günlük Scrum sırasında, üyeler bir önceki gün tamamlanan çalışmaları, bir sonraki gün için

planlanan çalışmaları ve bir sprint sırasında karşılaşılan sorunları tartışacaklar. Günlük stand-up toplantısı sırasında ekibin ilerlemesi izlenir.

- **Sprint İnceleme / Retrospektif:** Aynı zamanda scrum ustası tarafından barındırılır, yaklaşık 2-4 saat sürer ve takımın son sprintte neler başardığını ve hangi derslerin öğrenildiğini tartışır.

Scrum'da Test Cihazının Rolü



Scrum Sürecinde **Test Cihazının aktif bir rolü yoktur** . Genellikle test, Unit Test ile bir geliştirici tarafından gerçekleştirilir. Ürün sahibi de her sprint sırasında test sürecine sık sık dahil olur. **Bazı Scrum projelerinde, projenin doğasına ve karmaşıklığına bağlı olarak özel test ekipleri bulunur** .

Bir sonraki soru şudur: test kullanıcısı bir scrumda ne yapar? Aşağıdaki not cevaplayacak

Scrum'da Test Aktiviteleri

Test uzmanları, Scrum'ın çeşitli aşamalarında aşağıdaki etkinlikleri yapar:

Sprint Planlama

- Sprint planlamada, bir test uzmanı, ürün birikiminden test edilmesi gereken bir kullanıcı hikayesi seçmelidir.
- Bir test uzmanı olarak, seçilen kullanıcı hikayelerinin her biri için testi **bitirmek** için kaç saat (Efor Tahmini) alacağına karar vermelidir .
- Bir testçi olarak, sprint hedeflerinin ne olduğunu bilmelidir.
- Bir test uzmanı olarak önceliklendirme sürecine katkıda bulunun

Sprint

- Geliştiricileri birim testinde destekleyin
- Tamamlandığında kullanıcı hikayesini test edin. **Test yürütme,** hem test edenin hem de geliştiricinin el ele çalıştığı bir laboratuvarda **gerçekleştirilir** . Kusur, günlük olarak takip edilen Kusur Yönetimi aracına kaydedilir. Scrum toplantısı sırasında kusurlar tartışılabilir ve analiz edilebilir. Kusurlar **çözülür çözülmez** ve test için devreye alınır alınmaz yeniden test edilir
- Bir testçi olarak, konuşmak için tüm günlük standup toplantılarına katılır.
- Bir testçi olarak, mevcut sprintte tamamlanamayan herhangi bir iş yığını öğesini getirebilir ve bir sonraki sprint'e koyabilir.
- Tester, otomasyon betikleri geliştirmekten sorumludur. Sürekli Entegrasyon (CI) sistemi ile otomasyon testlerini planlar. Otomasyon, kısa teslimat süreleri nedeniyle önem kazanmaktadır. Test Otomasyonu, piyasada bulunan çeşitli açık kaynak veya ücretli araçlar kullanılarak gerçekleştirilebilir. Bu, test edilmesi gereken her şeyin kapsanmasını sağlamada etkili olduğunu kanıtlıyor. Yeterli Test kapsamı, ekip ile yakın bir iletişim ile sağlanabilir.

- CI otomasyon sonuçlarını inceleyin ve paydaşlara Raporlar gönderin
- Onaylanmış kullanıcı hikayeleri için işlevsel olmayan testlerin yürütülmesi
- Kabul Testleri için kabul kriterlerini belirlemek için müşteri ve ürün sahibi ile koordineli çalışın
- Sprint sonunda, test cihazı bazı durumlarda kabul testi (UAT) de yapar ve mevcut sprint için testin tamamlandığını onaylar.

Sprint Retrospektif

- Bir testçi olarak, mevcut sprintte neyin yanlış gittiğini ve neyin doğru gittiğini anlayacaktır.

- Bir test uzmanı olarak, öğrenilen dersi ve en iyi uygulamaları belirler

Test Raporlama

Scrum Test ölçümleri raporlaması, proje hakkında paydaşlara şeffaflık ve görünürlük sağlar. Bildirilen ölçümler, bir ekibin ilerlemesini analiz etmesine ve ürünü iyileştirmek için gelecekteki stratejisini planlamasına olanak tanır. Raporlamak için sıklıkla kullanılan iki metrik vardır.

Yakma tablosu: Scrum Master her gün, sprint için kalan tahmini işi kaydeder. Bu Burn Down Tablosundan başka bir şey değil. Günlük olarak güncellenir.

Yakma çizelgesi, projenin ilerleyişine hızlı bir genel bakış sağlar, bu çizelge, projedeki tamamlanması gereken toplam çalışma miktarı, her sprint sırasında tamamlanan iş miktarı gibi bilgileri içerir.

Hız geçmişi grafiği: Hız geçmişi grafiği , her sprintte ulaşılan takımın hızını tahmin eder. Bu bir çubuk grafiktir ve ekiplerin çıktısının zaman içinde nasıl değiştiğini gösterir.

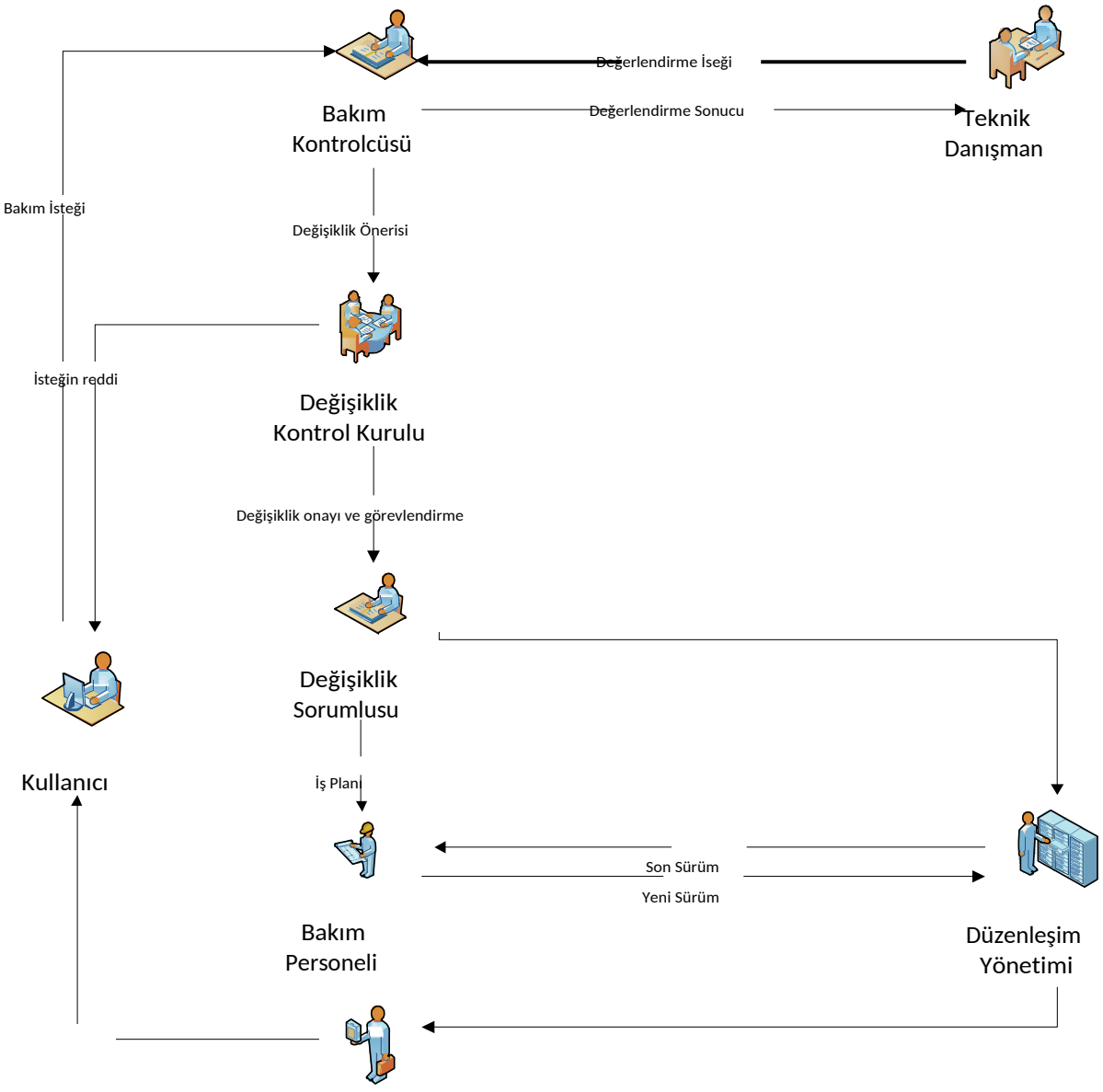
Yararlı olabilecek ek ölçümler, program yakma, bütçe yakma, tema tamamlanma yüzdesi, tamamlanan hikayeler - kalan hikayeler vb.

-

7.BAKIM

7.1 Giriş

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



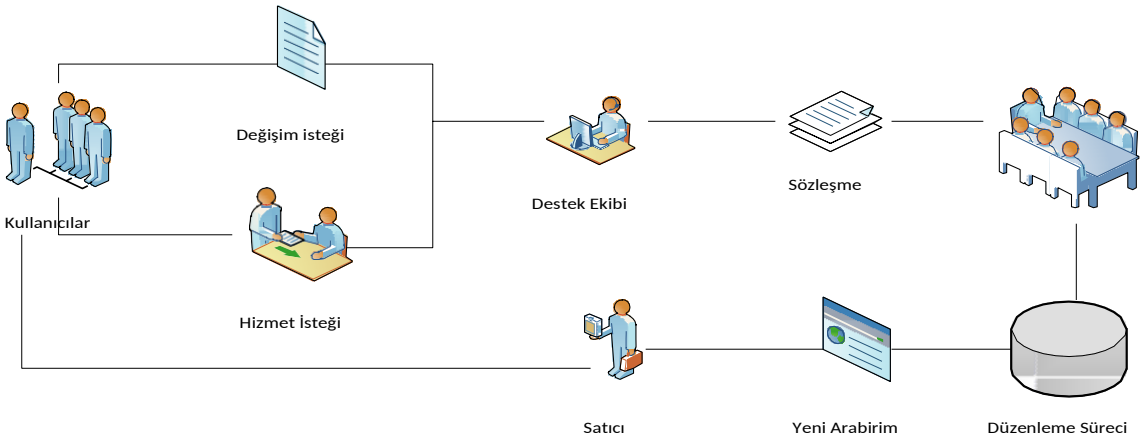
7.2 Kurulum

Sistem kurulumuna değinmek gerekirse sistem hazır olarak gönderilecek ve kolayca kurulumu yapılabilecektir.

7.3 Yerinde Destek Organizasyonu

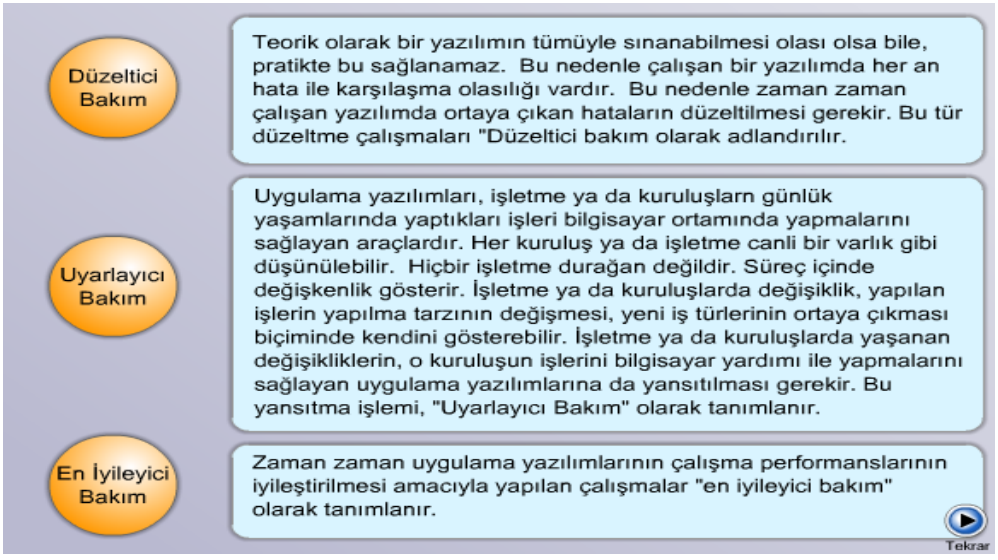
Yerinde destek için belirlediğimiz iletişim adresleri ile bize bildirimde bulunabileceklerdir.

7.4 Yazılım Bakımı



Şekil 7.2 Bakım Aşaması

7.4.1 Tanım



Şekil 7.3 Bakım Çeşitleri

7.4.2 Bakım Süreç Modeli

Projemizde uygulanan yazılım geliştirme modeli sayesinde bakım sürecine bakmak gerekirse bakım süreç modeli yukardaki yapılan işlemlerin tümünün baştan yapılması demektir.

8 SONUÇ

Sonuç olarak sistem hayata geçirildiği zaman neler değişeceği gözler önüne serdik. Bunun yanı sıra basit ama bir o kadarda etkili sistem sayesinde tez yazımı ile uğraşan ya da kontrol etmesiyle uğraşanların zaman bakımından kolaylık sağlayacaktır. Zaman kolaylığı ile birlikte tez yazım sürecini ya da kontrol sürecini kolaylaştıracaktır. Bu sistem sayesinde artık eskisi kadar yorulma olmayacaktır. Bu süreç boyunca bize destek olan herkese teşekkürlerimizi iletiyoruz.

9 KAYNAKLAR

1. https://www.researchgate.net/publication/324530793_Scrum_Software_Maintenance_Model_Efficient_Software_Maintenance_in_Agile_Methodology
2. https://www.scrum.org/forum/scrum-forum/16296/best-practices-testing-process-scrum
3. https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-TR.pdf
4. https://www.atlassian.com/agile/scrum#:~:text=Scrum%20is%20a%20framework%20that,and%20losses%20to%20continuously%20improve.
5. https://g.co/kgs/uNjk7L
6. http://sosbe.firat.edu.tr/tr/node/129