RESTORAN OTOMASYONU

FIRAT
ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
Yazılım Mühendisliği

Adı: Okan

Soyadı: Çavdar

Numarası: 15542518

İçindekiler

1.GİRİŞ

1.1 Projenin Amacı

Projenin amacı; Restoranlarda masa servislerinde kullanılabilecek şekilde tasarlanacak. Masaların durumları, ürün ekle-çıkar gibi işlemleri yapabilen bir otomasyon.

1.2 Projenin Kapsamı

Proje genellikle. Yazılım olarak masaüstü programı olacak. Masaüstü program kısmından personel masaları kontrol etmesi ürün ekleme gibi özellikler olacak.

1.3 Tanımlamalar ve Kısaltmalar

Müşteri: Masaya oturan kişi

Personel: Arka planda müşteri isteklerini takip eden, hazırlayan kişi.

2. PROJE PLANI

2.1 Giriş

Bu kapsamda öncelikle inceleme ve analiz yapılacak, bütün sistem bu inceleme ve analiz kısmına dayandırılacaktır. İnceleme kısmında projenin en ince detayına kadar müşteriden bilgi alınacak ve analiz edilecektir. Bundan sonra yapılabilirliğin hesabı yapılacak ve olumlu sonuçlanırsa projeye başlanacaktır.
Daha sonra izlenecek yolun belirlenmesi, kaynak ihtiyaçları ve gider tahmini, proje süresi ve yazılım geliştirmeye düşen görev ve işlemler analiz edilecek ve sonuçlandırılacaktır.

2.2 Projenin Plan Kapsamı

Teknik Karmaşıklık Tablosu

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

0: Hiçbir Etkisi Yok **1:** Çok Az etkisi var **2:** Etkisi Var **3:** Ortalama Etkisi Var

4: Önemli Etkisi Var **5:** Mutlaka Olmalı, Kaçınılamaz

Maliyet Kestirim Hesabı

Maliyet Kestirim Tablosu

Ölçüm parametresi	Sayı	Ağır	Toplam
Kullanıcı Girdi Sayısı	6	5	30
Kullanıcı Çıktı Sayısı	3	5	15
Kullanıcı Sorgu Sayısı	4	5	20
Kütük Sayısı	6	7	42
Dışsal Ara yüz Sayısı	5	5	25
Toplam Sayı			132

İşlev Noktası Hesaplama

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

İN = AİN x (0.65 x 0.01 x TKF)

127 x (0.65 x 0.01 x 58)=49,764

Tahmini oluşacak satır sayısı:

Satır Sayısı = İN *30

Satır Sayısı = 1492,92

Etkin Maliyet Modeli – COCOMO

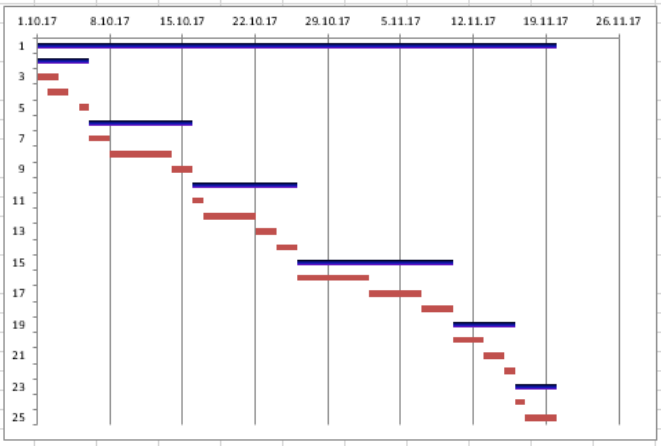
İş gücü (K) $K = a \times S^b$
Zaman (T) $T = c \times K^d$
a, b, c, d: her bir model için farklı olan katsayılar
S = bin türünden satır sayısı

İş gücü (K) $K = 2,4 \times 1^{1,05} = 2,4$ İş Gücü

Zaman (T) T = 1,12 x 1,56 Ay

2.3 Proje Zaman-İş Planı

Görev Adı	Süre (Gün)	Başlangıç	Bitiş
Restoran Otomasyonu	**50**	**1.10.2017**	**20.11.2017**
Proje İsterlerin Analizi	**5**	**1.10.2017**	**6.10.2017**
Projenin Tanımlanması	2	1.10.2017	2.10.2017
Sistem Mimarisinin Belirlenmesi	2	2.10.2017	4.10.2017
İster Analizinin Hazırlanması	1	5.10.2017	6.10.2017
Proje Planı	**10**	**6.10.2017**	**16.10.2017**
Proje Zaman İş Planı	2	6.10.2017	8.10.2017
Proje Ekip Yapısı Oluşturulması	6	8.10.2017	14.10.2017
Planların Oluşturulması	2	14.10.2017	16.10.2017
Sistem Çözümleme	**10**	**16.10.2017**	**26.10.2017**
Mevcut Sistemin İncelenmesi	1	16.10.2017	17.10.2017
Gereksenen Sistemin Mantıksal Modelleri	5	17.10.2017	22.10.2017
Arayüz Gerekleri	2	22.10.2017	24.10.2017
Belgeleme Gerekleri	2	24.10.2017	26.10.2017
Sistem Tasarımı	**15**	**26.10.2017**	**10.11.2017**
Genel Tasarım	7	26.10.2017	2.11.2017
Veri Tasarımı	5	2.11.2017	7.11.2017
Süreç Tasarımı	3	7.11.2017	10.11.2017
Sistem Gerçekleştirimi	**6**	**10.11.2017**	**16.11.2017**
Arayüz Gerçekleştirme	3	10.11.2017	13.11.2017
Veri Tabanı Gerçekleştirme	2	13.11.2017	15.11.2017
Kodlama	1	15.11.2017	16.11.2017
Doğrulama ve Geçerleme	**4**	**16.11.2017**	**20.11.2017**
Testlerin Belirlenmesi	1	16.11.2017	17.11.2017
Testlerin Gerçekleştirilmesi	3	17.11.2017	20.11.2017



2.4 Proje Ekip Yapısı

- Proje Yöneticisi

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

- Kalite Uzmanı

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

- Programcı

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

- Yazılım Destek Elemanı

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

- Donanım Ekip Lideri

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

- Donanım Mühendisi

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

- Donanım Destek Elemanı

Bilgisayar donanımlarında çıkan sorunları tespit edip engellemeye, düzeltmeye çalışır.

- Bilgisayar Ağ Uzmanı

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

- Sistem Çözümleyici

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

- Sistem Tasarımcı

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

- Sistem Yöneticisi

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

- Veri Tabanı Yöneticisi

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

2.5 Önerilen Sistemin Teknik Tanımları

Sistemde, masaüstü uygulamalar geliştirilecek. Bu uygulamalar personel tarafından kullanılacak ve kullanılabilirlik açısından olabildiğince basit tasarlanacaklardır.

Personelin masa takibi yapabilmesini sağlayan ve denetlenebilen bir masaüstü program geliştirilecektir.

2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları

İşletim Sistemleri	Programlama Dilleri	VTYS	Derleyiciler	Tasarım
• Windows	• C#	• Access	• .NET Framework	• Photoshop CS6

2.6.1 Photoshop

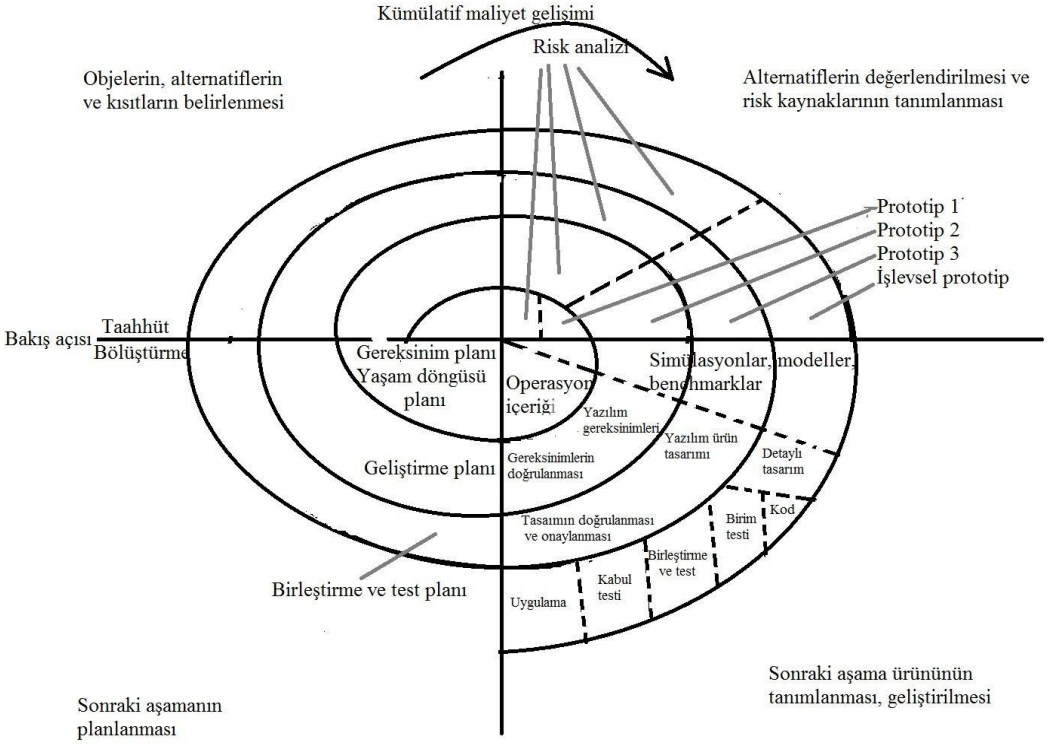
Masaüstü programda görselliğin oluşturulmasının kolay olması açısından Photoshop CS6'den faydalanılacaktır.

2.6.2 Access

Veri Tabanı olarak Access kullanılacak.

2.7 Proje Standartları, Yöntem ve Metodolojiler

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



Spiral Model

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

2.7.1 Spiral Modelin Yapısı

Planlama: -Amaç belirlenir,

-Alternatifler belirlenir,

-Kısıtlar belirlenir.

Risk Çözümleme: -Alternatifler değerlendirilir,

-Risk analizi yapılır.

Üretim: -Geliştirme yapılır,

-Bir sonraki ürün belirlenir.

Kullanıcı Değerlendirmesi:-Bir sonraki aşama planlandırılır,

-Kullanıcı değerlendirmeleri alınır.

2.7.2 Spiral Modelin Seçilme Sebepleri

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

1. Kullanıcı Katkısı

Üretim süreci boyunca ara ürün üretme ve üretilen ara ürünün kullanıcı tarafından sınanması temeline dayanır. Yazılımı kullanacak personelin sürece erken katılması

İlerde oluşabilecek istenmeyen durumları engeller.

2. Yönetici Bakışı

Gerek proje sahibi, gerekse yüklenici tarafındaki yöneticiler, çalışan yazılımlarla proje boyunca karşılaştıkları için daha kolay izleme ve hak ediş planlaması yapılır.

3. Yazılım Geliştirici (Mühendis) Bakışı

Yazılımın kodlanması ve sınanması daha erken başlar.

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

2.8.1 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:

1. 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.

2. Üretilen yazılım ve standartların bir program tarafından yapılır ve geliştirme projesine uygulanan standartlarla karşılaştırılırsa buna otomatikleştirilmiş yazılım değerlendirme adı verilecektir. Otomatikleştirilmiş değerlendirme bazı yazılım özelliklerinin nicel ölçümlerini gerektirir.

2.9 Eğitim Planı

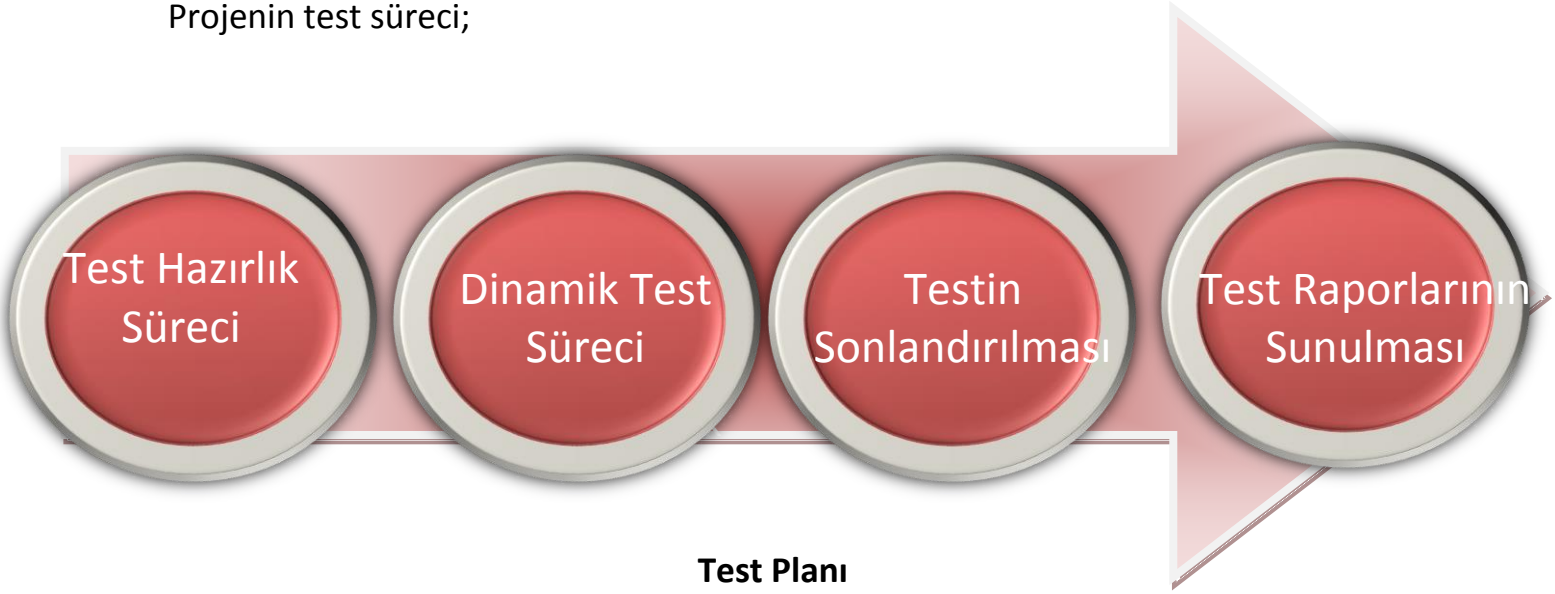
Sistemin nasıl kullanılacağı müşteriler tarafında basit olsa da personel ve kasiyer kısmında bir eğitim verilmelidir. Bu bölümde eğitimde neler yapılacağı ve planı yer alacaktır.

Projenin genel yapısı, çalışma sistemi, önbilgi	• Personel • Yönetici
Personel yazılımı eğitimi	• Personel • Yönetici
Kasiyer yazılımı eğitimi	• Personel

2.10 Test Planı

Helezonik modelde analiz, kodlama, test ve kullanım birbirini izler. Yani her bir prototip kullanıcıya verilmeden her seferinde test edilir. Bu testleri Yazılım Test Ekibi yapacaktır.

Kodlamanın bitmesi beklenmeden test hazırlık sürecine geçilecektir. Projenin test süreci;



Test Planı

Test Hazırlık Süreci

Testin sağlıklı geçebilmesi için önceden planlanmış bir test planına ihtiyaç vardır. Bu süreçte bu test planı hazırlanacaktır. Test hazırlık sürecinde şu standartlara uyulacaktır:

- Öncelikle test edilecek projenin analiz ve teknik tasarım aşamaları ile ilgili dökümanlar, test ekibi tarafından incelenecektir.
- Yazılım içinde test edilecek ve edilmeyecek modüller belirlenecektir.
- Risk analizi yapılır ve yapılan değerlendirmeye göre dinamik test aşamasında uygulanacak olan test teknikleri ve metodları belirlenir.
- Dinamik testin uygulanacağı ortamlar ve bu ortamların ihtiyaçları belirlenip, uygun şartlar sağlanacaktır.
- Test ekibi içinde görev paylaşımı ve zaman planlaması yapılır.
- Testin sonlandırma kriterleri belirlenir.
- Bir programa belirli girdiler (input) verildiğinde hangi çıkışların (output) ne şekilde alınması gerektiğini bildiren test case senoryaları belirlenir.
- Dinamik testin hangi adımlarla ve ne şekilde uygulanacağının belirtildiği test planı hazırlanır.

Dinamik Test Süreci

Bu süreç kodlama çalışmalarının bitmesine yakın bir dönemde başlayacaktır. Bulunan tüm hatalar çözülmeden ve testin sonlandırma kriterleri sağlanmadan sona ermez. Test edilecek yazılımın türüne göre, dinamik olarak uygulanacak test teknikleri ve bu tekniklerin uygulanma metotları farklılık gösterebilir.

Testin Sonlandırılması

Yapılan testler sonucunda bulunan hatalar düzeltildikten sonra test sonlandırma kriterleri (test hazırlık süreci) kontrol edilir. Eğer tüm kriterlerin kabul edilebilir düzeyde sağlandığı tespit edilirse test sonlandırılır. Testin sonlandırılmasının ardından uygulama müşteri testine açılır (Kullanıcı Kabul Testi). Müşterilerin bulduğu hatalar veya değiştirilmesi istenilen noktalar gözden geçirilerek tekrar test ekibinin kontrolüne sunulur. Bu kontrolden çıkan uygulama ürün aşamasına geçer ve böylelikle yazılım test süreci sona erdirilerek, yazılım geliştirme sürecinin son basamağına geçilmiş olunur.

Test Raporlarının Sunulması

Bu süreçte testler yapıldıktan sonra raporlanıp yetkili kişiye verilecektir.

2.11 Bakım Planı

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

Müşteri geri dönüşlerinde olumsuz bir durum var mı?
Sistemin daha da iyileştirilmesi için neler yapılabilir?
Sistem loglarında ciddi bir problem gözlenmiş mi?
Personel kullandığı programlarda bir problem yaşamış mı?
Sistem yeterince hızlı çalışıyor mu?

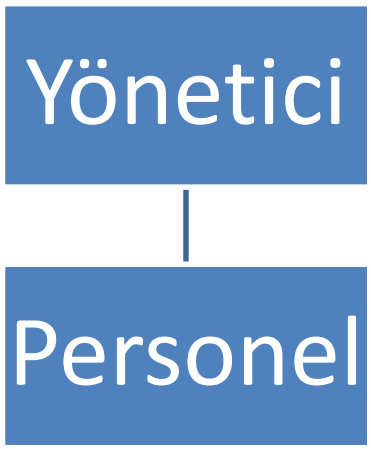
Aylık Bakım Planı

3.SİSTEM ÇÖZÜMLEME

3.1 Mevcut Sistemin İncelenmesi

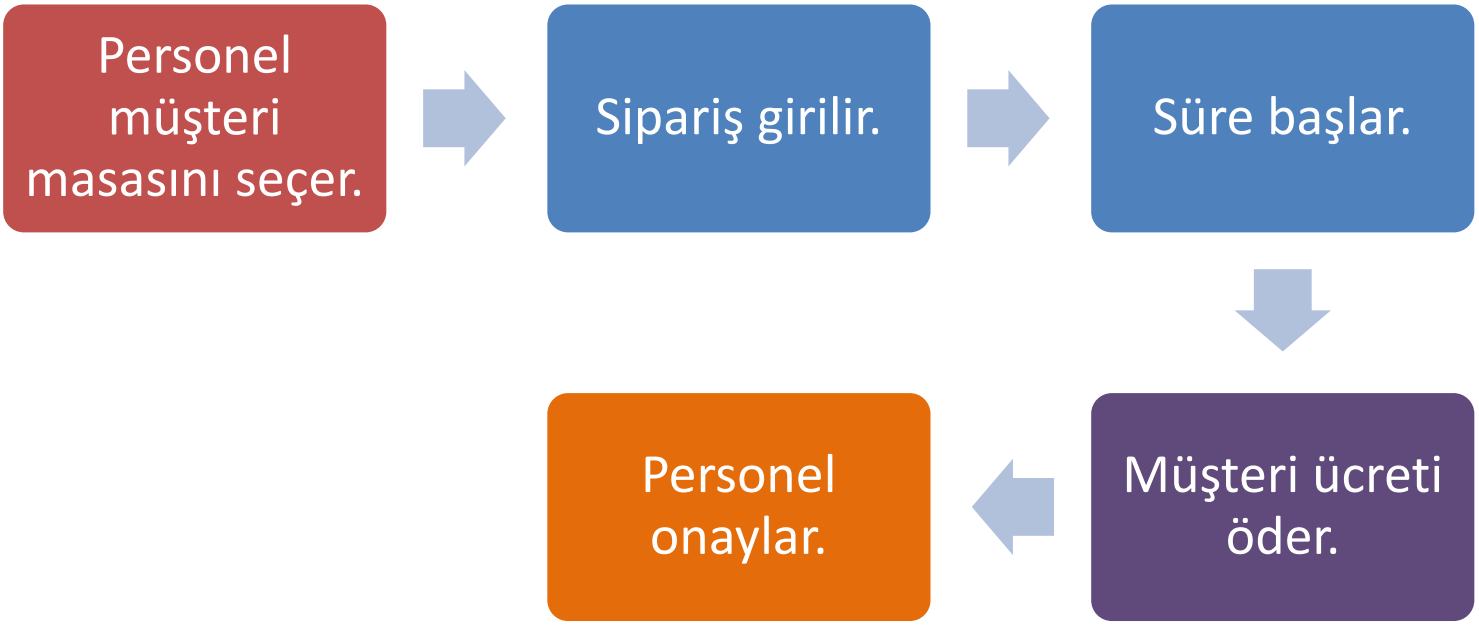
Projenin gereksinimlerin araştırılması, tanımlanması, ortaya çıkarılması ve düzgün bir şekilde, terimsel olarak açıklanması bu bölümde yapılacaktır.

3.1.1 Örgüt Yapısı



Örgüt Yapısı

3.1.2 İşlevsel Modeli



İşlevsel Model

3.1.3 Varolan Yazılım/Donanım Kaynakları

3.1.3.1 Microsoft Office Excel

Gantt Diyagramı çizilmesinde faydalanılmıştır.

3.1.3.2 Microsoft Word

Dokümantasyonun hazırlanmasında faydalanılmıştır.

3.1.3.3 Microsoft Visio

Diyagramların çiziminde faydalanılmıştır.

3.1.3.4 Adobe Photoshop CS6

Belirli resimlerin çizilmesinde, düzenlenmesinde faydalanılmıştır.

3.1.5 Varolan Sistemin Değerlendirilmesi

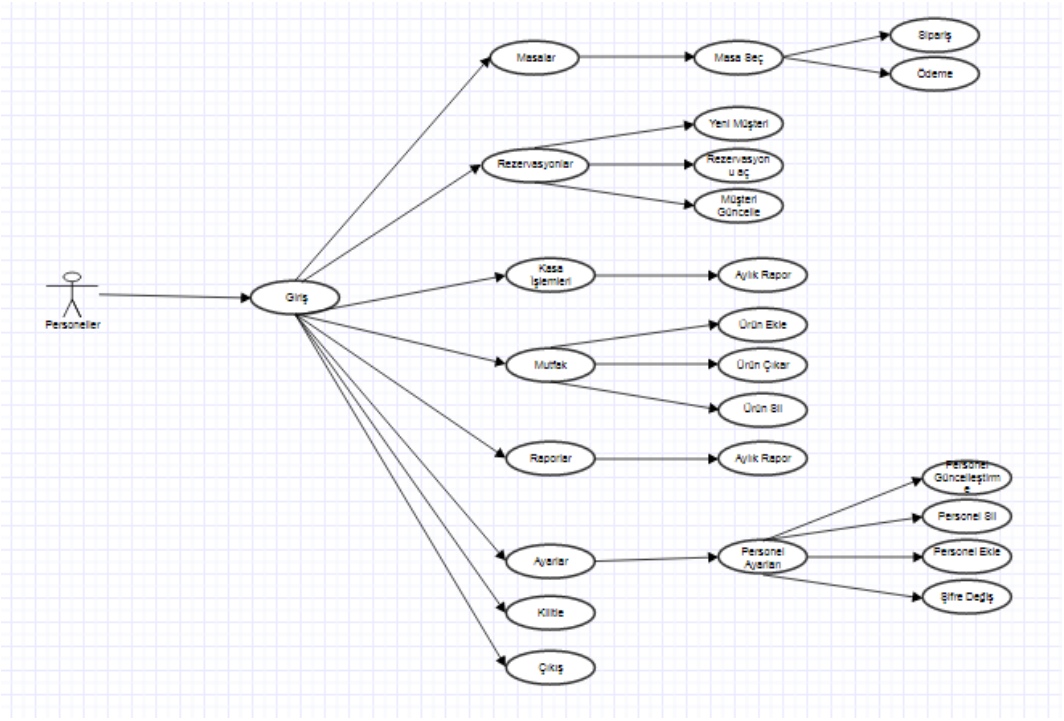
Piyasada tam olarak projemdeki gibi bir sistem mevcut değil. Basit QR Kod okuyucu sistemler incelenmiştir.

3.2 Gereksenen Sistemin Mantıksal Modeli

3.2.1 Giriş

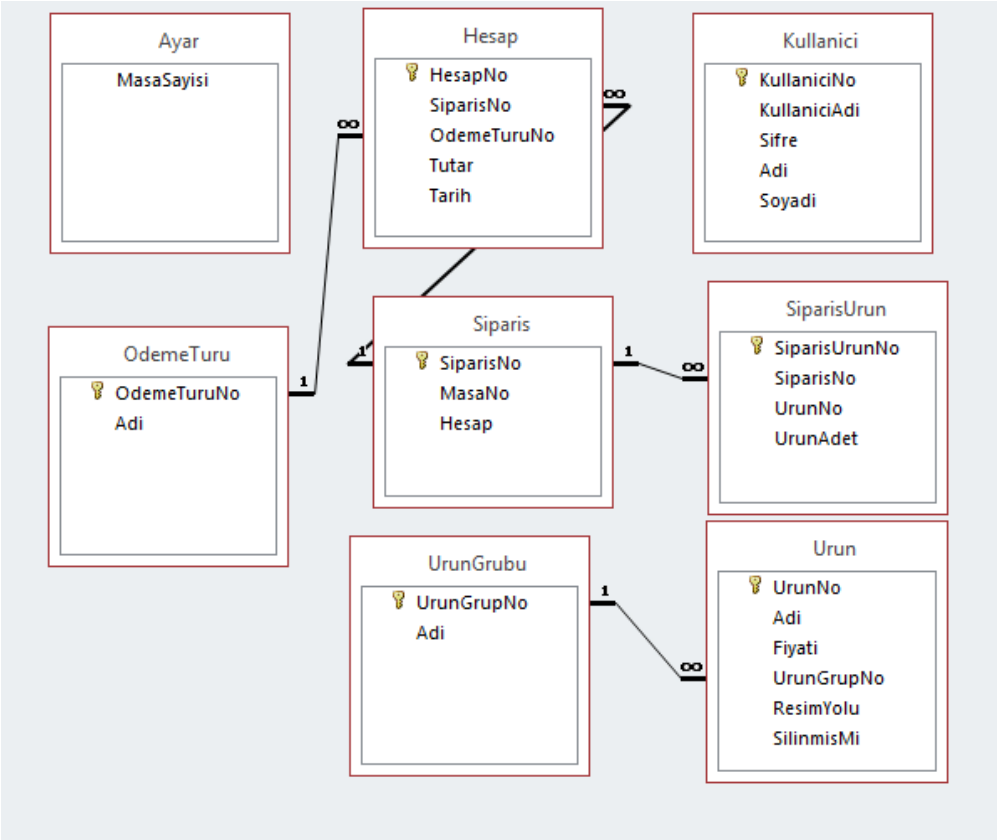
Bu bölümde önerilen sistemin işlevsel yapısı, veri yapısı ve kullanıcı ara yüzünde çözümleme yapılır. Bu model daha çok bilgi sistemini geliştirecek teknik personele yöneliktir. Mantıksal model olarak da tanımlanır.

3.2.2 İşlevsel Model



Use Case Diyagramı (Genel)

3.2.3 Veri Modeli



Veri Modeli

3.2.4 Veri Sözlüğü

Ayar Tablosu	Veri Tipi	Açıklama
MasaSayisi	İnt	Restoran masa sayısı.

Hesap Tablosu	Veri Tipi	Açıklama
HesapNo	İnt – Primary key	Bakiye tablosunun birincil anahtarıdır.
SiparisNo	int	Sipariş tablosundaki primary key ile ilişkili.
OdemeTuruNo	int	Ödemek türü tablosundaki primary key ile ilişkili.
Tutar	Money	Sahip olunan bakiye miktarı tutulacak.
Tarih	Datetime	Tarih bilgisi tutulacak.

Kullanıcı Tablosu	Veri Tipi	Açıklama
KullaniciNo	İnt – Primary key	Kullanıcı tablosunun birincil anahtarıdır.
KullaniciAdi	Nvarchar(15)	Kullanıcı adı tutulacak.
Sifre	Nvarchar(15)	Kullanıcı şifresi tutulacak.
Adi	Nvarchar(25)	Kullanıcının adı tutulacak.
Soyadi	Nvarchar(25)	Kullanıcının soyadı tutulacak.

Ödeme Türü Tablosu	Veri Tipi	Açıklama
OdemeTuruNo	İnt – Primary key	Ödeme Türü tablosunun birincil anahtarıdır.
Adi	Nvarchar(10)	Ödeme türü tutulacak.

Sipariş Tablosu	Veri Tipi	Açıklama
SiparisNo	İnt – Primary key	Sipariş tablosunun birincil anahtarıdır.
MasaNo	int	Masa numarası tutulacak.
Hesap	bit	Hesabın ödenip ödenmediğinin bilgisi tutulacak.

Sipariş Ürün Tablosu	Veri Tipi	Açıklama
SiparisUrunNo	İnt – Primary key	Sipariş Ürün tablosunun birincil anahtarıdır.
Sipariş No	int	Sipariş numarası tutulacak.
UrunNo	int	Ürün tablosundaki primary key ile ilişkili.
UrunAdet	İnt	Ürün adeti tutalacak.

Ürün Tablosu	Veri Tipi	Açıklama
UrunNo	İnt – Primary key	Ürün tablosunun birincil anahtarıdır.
Adi	int	Ürün adı tutulacak.
Fiyati	money	Ürün fiyatı tutulacak.
UrunGrupNo	İnt	Ürün Grubu tablosundaki primary key ile ilişkili.
ResimYolu	Nvarchar(100)	Ürün resim konum bilgisi tutulacak.
SilinmisMi	Bit	Ürünün silinme durumu tutulacak.

Urun GrubuTablosu	Veri Tipi	Açıklama
UrunGrupNo	İnt – Primary key	Ürün grubu tablosunun birincil anahtarıdır.
Adi	Nvarchar(10)	Ürün grup bilgisi tutulacak.

3.2.5 Başarım Gerekleri

- Sistem hızlı çalışmalı
- Ara yüz basit olmalı
- Personel ekip yapısı düzgün seçilmeli
- Personel eğitimi yapılmalı
- Güvenlik sorunları çıkmamalı

4.SİSTEM TASARIMI

4.1 Genel Tasarım Bilgileri

4.1.1 Genel Sistem Tanımı

Gerekenler :
Masaüstü uygulamalardan ulaşılabilecek bir sunucu

Masaüstü uygulamalar
Bu uygulamalar veritabanıyla bağlantılı olacak.

Masa üzerinde işlemler yapılabilecek.
Personel masaya sipariş girebilecek.

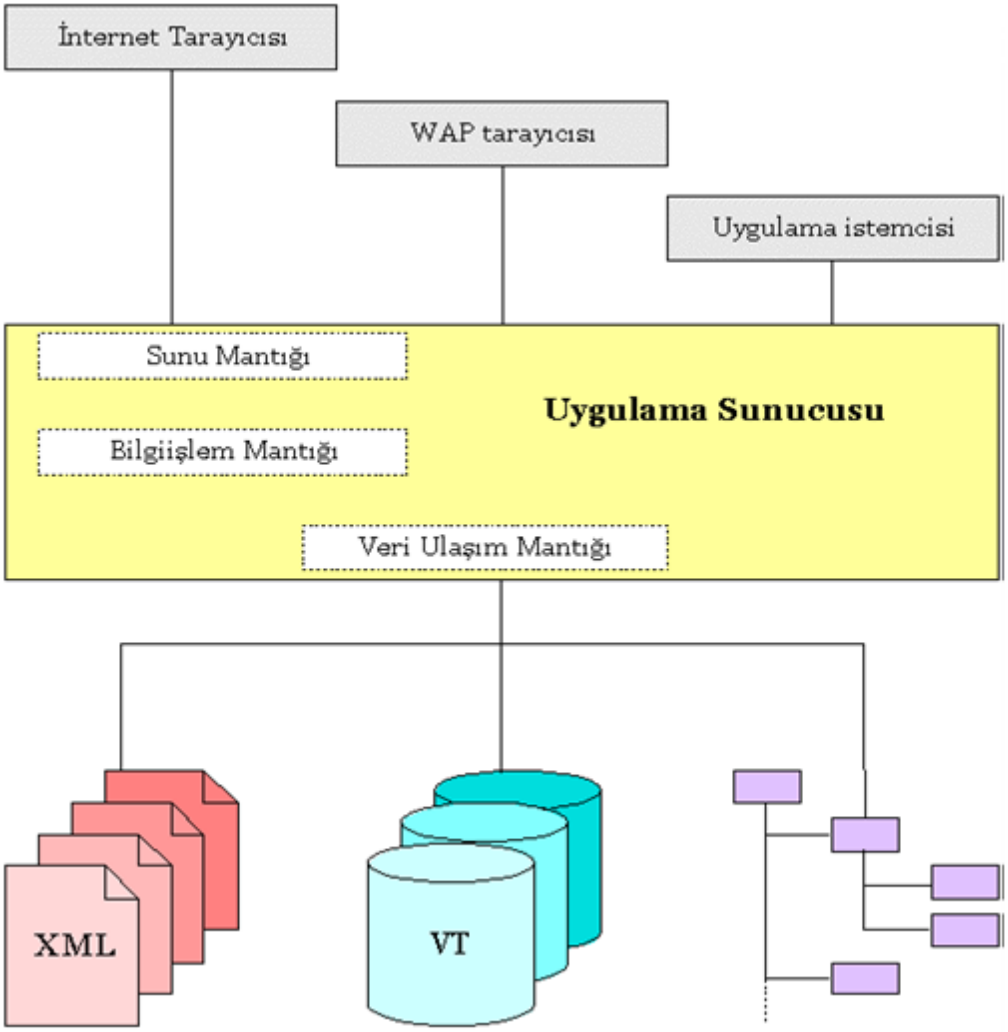
Hesap ödeme tipi seçenekleri çıkacak.
Ödeme tipleri online kredi kartı veya peşin ödemeler

4.1.2 Varsayımlar ve Kısaltmalar

Müşteri: Masaya oturan kişi

Personel: Arka planda müşteri isteklerini takip eden, hazırlayan kişi.

4.1.3 Sistem Mimarisi



Sistem Mimarisi

5 katmandan oluşmaktadır;

İstemci Katı (Client Tier): Bu kat, geliştirilen uygulamaya ya da sisteme bağlanan diğer uygulamalar ya da cihazlardan oluşur. Örneğin: İnternet tarayıcısı, Java applet, WAP telefon...

Sunuş Katı (Presentation Tier): Bu kat, sistemin istemcileri için gerekli olan her türlü sunuş mantığını içinde bulundurur. Uygulamaya bağlanan istemcilerin taleplerini kaydeder, gerekli iş matığının uygulanmasını sağlar, talebin işlenmesi sonucu ortaya çıkan veriyi sunulur hale getirip istemciye cevap yollar. J2EE'yi oluşturan teknolojilerden ikisi JSP (JavaServer Pages) ve Java Servlet bu katta bulunur.

Uygulama ya da İş Katı (Application/Business Tier): Uygulamanın hedef aldığı ve gereklerini tatmin etmek için geliştirildiği işe dayalı tüm bilgiişlem bu katta

toplanır. Bu görev, J2EE'yi oluşturan bir diğer teknoloji olan EJB (Enterprise JavaBeans)'ler tarafından sağlanır.

Entegrasyon Katı (Integration Tier): Bu kat, uygulamanın görevini yerine getirmesi için gerekli olan sistem dışı yazılımlara, sistemlere ya da veri tabanlarına bağlantıları sağlamakla yükümlüdür. J2EE uygulamalarının bu bölümleri, genelde, JDBC (Java DataBase Connectivity), J2EE Connector ya da bağlantı kurulan yazılımlara özel arayüzleri kullanırlar.
Kaynak Katı (Resource Tier): Bilgiişlem için gerekli veriler ve dış servisler bu katı oluşturur.

4.1.4 Testler

Yazılım Testi

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

Yeterlilik Testi

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

Sistem Testi

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

4.2 Ortak Alt Sistemlerin Tasarımı

4.2.1 Ortak Alt Sistemler

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

Yetkilendirme Alt Sistemi,

Güvenlik Alt Sistemi,

Yedekleme Alt Sistemi,

Veri Transferi Alt Sistemi,

Arşiv Alt Sistemi,

Dönüştürme Alt Sistemi.

4.2.1.1 Yetkilendirme Alt Sistemi

Özellikle kurumsal uygulamalarda farklı kullanıcıların kullanabilecekleri ve kullanamayacakları özellikleri ifade eder.

İşlev bazında yetkilendirme

Ekran bazında yetkilendirme

Ekran alanları bazında yetkilendirme

Oracle veri tabanına erişim konusunda yetkilendirme yapmaktadır.

4.2.1.2 Güvenlik Alt Sistemi

Yapılan bir işlemde, işlemi yapan kullanıcının izlerinin saklanması gerekmektedir. Bunlar LOG files(Sistem günlüğü) dosyalarında tutulmalıdır.

4.2.1.3 Yedekleme Alt Sistemi

Her bilgi sisteminin olağandışı durumlara hazırlıklı olmak amacıyla kullandıkları veri tabanı (sistem) yedekleme ve yedekten geri alma işlemlerinin olması gerekmektedir. Bu yüzden veri tabanı yedeklemesine önem verilmelidir.

4.2.1.4 Veri İletişim Alt Sistem*i*

Coğrafi olarak dağıtılmış hizmet birimlerinde çalışan makineler arasında veri akışının sağlanması işlemleridir. Bu veri iletişimi 2 türlü sağlanmaktadır.
Çevrim içi veri iletimi (real time)
Çevirim dışı veri iletimi (disketler, teypler)

4.2.1.5 Arşiv Alt Sistemi

Belirli bir süre sonrasında sık olarak kullanılmayacak olan bilgilerin ayrılması ve gerektiğinde bu bilgilere erişimi sağlayan alt sistemlerdir.
Bunun için aktif veri tabanı dışında arşiv tutacak bir veri tabanı gerekmektedir.

4.2.1.6 Dönüştürme Alt Sistemi

Geliştirilen bilgi sisteminin uygulamaya alınmadan önce veri dönüştürme (mevcut sistemdeki verilerin yeni bilgi sistemine aktarılması) işlemlerine ihtiyaç vardır.

5.SİSTEM GERÇEKLEŞTİRİMİ

5.1 Giriş

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

Söz konusu ortam, kullanılacak programlama dili ve yazılım geliştirme araçlarını içerir. Söz konusu ortamda belirli bir standartta geliştirilen programlar, gözden geçirilir, sınanır ve uygulamaya hazır hale getirilir. Üretilen kaynak kodların belirlenecek bir standartta üretilmesi yazılımın daha sonraki aşamalardaki bakımı açısından çok önemlidir. Tersi durumda kaynak kodların okunabilirliği, düzeltilebilirliği zorlaşır ve yazılımın işletimi süresince ortaya çıkabilecek sorunlar kolayca çözülemez.

5.2 Yazılım Geliştirme Ortamları

5.2.1 Programlama Dilleri

Yazılım geliştirilirken pek çok farklı programlama dili kullanılmıştır. Bunlar;

- C#

Bu dilleri kısaca açıklamak gerekirse

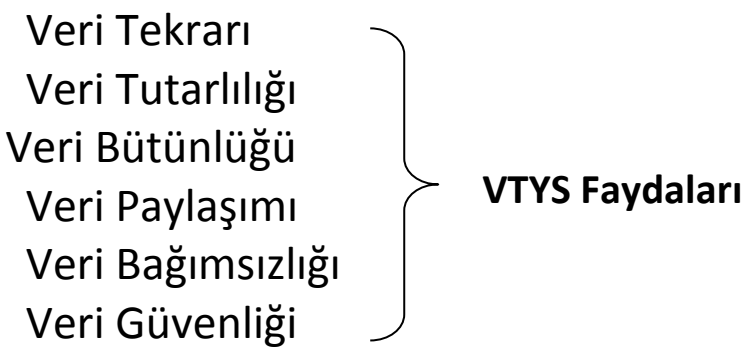
C#, yazılım sektörü içerisinde en sık kullanılan iki yazılım dili olan C ve C++ etkileşimi ile türetilmiştir. Ayrıca C#, ortak platformlarda taşınabilir bir (portable language) programlama dili olan Java ile pek çok açıdan benzerlik taşımaktadır. En büyük özelliği ise .Net Framework platformu için hazırlanmış tamamen nesne yönelimli bir yazılım dilidir. Yani nesneler önceden sınıflar halinde yazılıdır. Programcıya sadece o nesneyi sürüklemek ve sonrasında nesneyi amaca uygun çalıştıracak kod satırlarını yazmak kalır.

Microsoft tarafından geliştirilen C#, C++ ve Visual Basic dillerinde yer alan tutarsızlıkları kaldırmak için geliştirilmiş bir dil olmasına rağmen kısa süre içerisinde nesne yönelimli dillerin içinde en gelişmiş programlama dillerinden biri olmayı başarmıştır.

5.2.2 Veri Tabanı Yönetim Sistemleri

Veri tabanı yönetim sistemi (**VTYS**, İngilizce: *Database Management System*, kısaca **DBMS**), veri tabanlarını tanımlamak, yaratmak, kullanmak, değiştirmek ve veri tabanı sistemleri ile ilgili her türlü işletimsel gereksinimleri karşılamak için tasarlanmış sistem ve yazılımdır.

5.2.2.1 VTYS Kullanımının Ek Yararları



Neden VTYS?

- Veri tutarlılığının sağlanması

Faydaları: Verilerin farklı tablolarda değişik değerler almasının önlenmesi

- Veri paylaşımının sağlanması

Faydaları: İnsan kaynaklarının ve donanımın verimli kullanılması

- Veri tekrarının azaltılması (data redundancy)

Faydaları: Donanım harcamalarının azalması, tutarsızlığın önlenmesi

- Verilerin güvenliğinin sağlanması (data security)

Faydaları: Yetki-sorumluluk bazında gerekli verileri görebilme, yetkisiz kişilerin sisteme girememesi, yedekleme hatadan kurtarma (recovery)

5.2.2.1 Veri Tabanı Dilleri ve Arabirimleri

Bu proje VTYS olduğundan Access dili kullanılacaktır. **Access**, İlişkisel Veri Tabanı Yönetim Sistemi ile çalışan bir veri tabanı oluşturma programıdır. İlişkisel Veri Tabanı Yönetim Sistemi sisteminde bir veri tabanı dosyasında birden fazla tablo oluşturulabilir ve bu tablolar arasında birbirleriyle ilişki kurulabilir. Kurulan ilişkiler sayesinde farklı tablolardaki veriler sanki aynı tablodaymış gibi kullanılabilir.

Microsoft Access bir İlişkisel Veri Tabanı Yönetim Sistemi uygulamasıdır. Bir veri tabanını oluşturmak ve kullanmak Access ile diğer veri tabanı uygulamalarına göre çok daha kolaydır. Bunun nedeni Access'in, Windows ortamının Grafiksel Kullanıcı Arabiriminin sağladığı avantajların tümünden

yararlanma imkânı vermesidir. Grafiksel Kullanıcı Arabirimi, karmaşık komut dizilerini öğrenmeyi gerektirmeden, ekran üzerindeki nesneler ve simgeler yardımıyla, fare desteğinden de yararlanarak kullanıcının çalışmasına olanak verir. Örneğin -, geleneksel veri tabanı uygulamalarında iki tablo arasında bağlantı kurmak için oldukça karmaşık komut dizileri yazmak gerekirken, Access'te bu iş basit bir fare hareketiyle gerçekleştirilebilir.

5.2.2.3 Veri Tabanı Sistem Ortamı

VTYS Mimarisi İçsel Yüzey, Kavramsal Yüzey, Dışsal Yüzey olarak ayrılır. Kısaca bu yüzeylerde hedeflenen amaçlar;

İçsel Yüzey

•Fiziksel veri yolunu detaylarıyla belirtir.

Kavramsal Düzey

•Kavramsal şema içerir ve kullanıcılar için veri tabanın yapısına açıklar.

Dışsal Yüzey

•Bu yüzey kullanıcı görüşlerini içerir. Her şema bir veri tabanının bir bölümünü açıklar.

5.2.2.4 VTYS'nin Sınıflandırılması

VTYS olarak ilişkisel veri modeli kullanılmıştır. İlişkisel veri tabanının kullanım nedeni birbiriyle alakalı aynı türden verilerin kullanılacağı için ayrıca projede kullanılacak en iyi alt yapı olduğu için ilişkisel veri tabanı seçilmiştir. Veri Tabanı Yöneticisinin Görevleri:

• Veriler üzerinde yapılacak uygulama gereksinimlerini belirlemek, veri tabanı içeriğini oluşturmak, veri tabanı şemalarını (tabloları) tanımlamak.

• Bütünlük kısıtlamalarını (Primary Key, Foreign Key, Unique, Check, Not Null) belirleyip tanımlamak.

• Veri tabanı kullanıcılarını ve her kullanıcının hangi veriler üzerinde hangi işlemleri yapmaya yetkili olduğunu belirlemek; kullanıcı ve kullanım yetkilerini tanımlamak.

• Veri Tabanı Yönetim Sisteminin sunduğu seçenekler çerçevesinde, veri tabanının fiziksel yapısı ile ilgili parametreleri ve erişim yollarını (dizinleri) belirlemek ve tanımlamak.

• Yedekleme, yeniden başlatma ve kurtarma düzenlerini belirlemek.

• Veri tabanı sistemini sahiplenmek, işletimini izlemek, veri tabanının sürekli olarak kullanıma açık olmasını sağlamak.

• Gereksinimlerdeki değişiklikleri izlemek ve değişikliklere paralel olarak veri tabanı içeriği, şema tanımları, bütünlük kısıtlamaları, fiziksel yapı ile ilgili parametreler, erişim yolları, kullanıcılar ve kullanıcı yetkilerinde gerekli değişiklikleri oluşturmak ve tanımlamak.

• Veri tabanı bütünlük kısıtlamalarının yeterliliğini izlemek; bütünlük kısıtlamaları ile ilgili gerekli değişiklikleri oluşturmak ve tasarlamak.

• Veri tabanı kullanım istatistiklerini ve veri tabanı başarımını izlemek; varsa sorunları ve yetersizlikleri belirlemek ve gerekli her türlü önlemi almak.

5.3 CASE Araç ve Ortamları

Microsoft Office Excel: Gantt Diyagramı çizilmesinde faydalanılmıştır.

Microsoft Word: Dokümantasyonun hazırlanmasında faydalanılmıştır.

Microsoft Visio: Diyagramların çiziminde faydalanılmıştır.

Adobe Photoshop CS6: belirli resimlerin çizilmesinde, düzenlenmesinde faydalanılmıştır.

5.4 Kodlama Stili

5.4.1 Açıklama Satırları

Bir yazılım geliştirirken kodların tekrar kullanılabilmesi, başkaları tarafından anlaşılabilmesi için kodlara açıklama satırları koyulur. Bunlar genel olarak

Bir paragraf şeklindeyse;

/*Açıklama
Açıklama
Açıklama
*/

Tek bir satır ise;
//Açıklama
Şeklinde ifade edilir.

5.4.2 Kod Biçimlemesi

Kod biçimlenmesi açıklama satırlarına olan ihtiyacı azaltır. Kod biçimlemesinde önemli olan az satır değil kodun okunabilirliğidir. Bu projede bu kriterler göze alınmalıdır.

5.4.3 Anlamlı İsimlendirme

Kodların okunabilirliğini ve anlaşılabilirliğini sağlayan önemli unsurlardan biri de kullanılan ve kullanıcı tarafından belirlenen belirteçlerin (Değişken adları, kütük adları, Veri tabanı tablo adları, işlev adları, yordam adları vb.) anlamlı olarak isimlendirilmesidir.

5.4.5 Yapısal Programlama Yapıları

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

Üç temel Yapısal Programlama Yapısı bulunmaktadır:

- Ardışıl işlem yapıları
- Koşullu işlem yapıları
- Döngü yapıları

5.5 Olağan Dışı Durum Çözümleme

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

5.6 Kod Gözden Geçirme

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

5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular

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

1. Kod doğru bir şekilde build edildi mi? Kaynak kod derlendiğinde hata olmamalı. Kodda yapılan değişikliklerde uyarılar(warning) olmamalı.

2. Kod çalıştırıldığında beklendiği gibi davrandı mı?

3. Kodun sadece çalışırlığına bakılmamalı, kod tasarımı da göz önün de bulundurulmalıdır. Optimizasyon için öneriler takım olarak değerlendirilmelidir.

4. Gözden geçirilen kod anlaşılıyor mu? Gözden geçiricinin kodu anlaması gerekir. Eğer anlaşılmadıysa, gözden geçirme tamamlanmış olmaz veya kod iyi yorumlanabilmiş sayılmaz.

5. Geleneksel kodlama standartlarına uyuldu mu? Değişken isimlendirme, satır başı boşluklar, parantez stilleri vs. takip edilmeli.

6. Telif hakkı bilgisi ve uygun bir başlıkla başlayan kaynak dosya var mı? Her bir kaynak dosyası bu bilgilerle başlamalı, bütün kaynak dosyaları, fonksiyonelliğini anlatan bir dosya içermelidir.

7. Değişken deklarasyonlarına yorum satırları eklenmiş mi? Yorumlar, değişkenlerin görevlerini açıklaması gerekir. Özellikle her bir global değişkenin amacı ve neden global olarak tanımlandığı belirtilmelidir.

8. Sayısal verilerin birimleri açıkça belirtilmiş mi? Sayısal verilerin birimleri yorum satırı olarak belirtilmeli. Örneğin, eğer bir sayı uzunluğu temsil ediyorsa, metre mi feet mi olduğu gösterilmelidir.

9. Bütün fonksiyonlar, metotlar ve classlar dokümante edilmiş mi? Her bir fonksiyon, metot ve class'ın tanımlanmasının üstünde bir iki cümle ile açıklaması yer almalıdır. Amacı vurgulamalı ve tasarım gerekliliklerini işaret etmelidir.

10. Fonksiyonların kullandığı input ve output parametreleri açıkça tanımlandı mı?

11. Karmaşık algoritmalar ve kod optimizasyonları yeterli olacak şekilde açıklanmış mı? Karmaşık alanlar, algoritmalar ve kod optimizasyonları için yeterince yorum satırı eklenmelidir. Öyle ki, diğer geliştiriciler kodu anlayabilmeli ve kalınan yerden devam ettirebilmelidir.

12. Kodun çeşitli yerlerinde yorum satırlarıyla açıklamalar var mı? Kodun çeşitli yerlerinde yorum satırlarıyla açıklamalar olmalı. "Ölü Kod"lar çıkarılmalı. Eğer geçici bir kod bloğu ise neden tanımlandığı belirtilmeli.

13. Koddaki eksik işlevsellikler veya çözümlenmemiş sorunlar yorum satırlarında ifade edilmiş mi? Bu ifadeler eksikleri ve yapılacakları açıklamalıdır.

Sonradan arandığında bulunabilmesi için de ayırıcı bir işaretleyici kullanılmalı, örneğin TODO.

14. Her zaman bir fonksiyonun döndürebileceği hatalar düzün bir şekilde handle edilmeli. Fonksiyonun üreteceği her bir sonuç düşünülmeli, her durum kontrol edilmeli ve kodun kalan kısmının yürütülmesini etkileyen hatalar yakalanmış olmalıdır.

15. Alınan hatalardan sonra tüm kaynaklar ve hafıza temizlenip serbest bırakılıyor mu? Bundan emin olunmalı. Bir hata meydana geldiğinde, dosya, soket ve veritabanı bağlantı objeleri gibi tüm objeler dispose edilmeli.

16. Exception'lar uygun bir şekilde yakalanıyor mu? Eğer fırlatılan exception kodun devamında kullanılıyorsa, bu fonksiyon devamında düzgün bir şekilde handle edilmeli veya yakalanmalı.

17. Tüm global değişkenler thread-safe olmalı. Eğer global değişkenlere birden fazla thread ile erişiliyorsa, kodun çalışmasını değiştirebilir veya sekronizasyon mekanizmasını engelleyebilir. Yine benzer bir şekilde olarak bir veya daha fazla thread ile erişilen objeler varsa, üyeler korunmalıdır.

18. Tespit edilen hata kodun başka yerlerini de etkiliyor mu, kontrol edilmeli? Hatanın tüm ekranlarda giderildiğinden emin olunmalı.

19. Kodun değişiklik yapılan yerlerinde, eski halini ve neden yapıldığını mutlaka açıklama olarak eklenmelidir.

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

5.6.2.1 Öbek Arayüzü

- Her öbek tek bir işlevsel amacı yerin 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

- 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 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 açıklamaları var mı?

5.6.2.3 Veri Kullanımı

- 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

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

5.6.2.5 Sunuş

- Her satır, en fazla bir deyim içeriyor mu?
- Bir 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

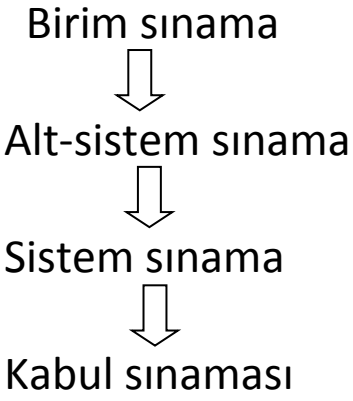
6.1 Giriş

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

6.2 Sınama Kavramları

Sınama ve Bütünleştirme işlemlerinin bir strateji içinde gerçekleştirilmesi, planlanması ve tekniklerinin seçilmesi gerekmektedir.

Sınama işlemleri dört ana sınıfta incelenebilir:



6.2.1 Birim Sınama

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

6.2.2 Alt-Sistem Sınama

Alt-sistemler modüllerin bütünleştirilmeleri ile ortaya çıkarlar.
Yine bağımsız olarak sınamaları yapılmalıdır.

Bu aşamada en çok hata arayüzlerde bulunmaktadır. Bu yüzden arayüz hatalarına doğru yoğunlaşılmalıdır.

6.2.3 Sistem Sınaması

Üst düzeyde, bileşenlerin sistem ile olan etkileşiminde çıkacak hatalar aranmaktadır.

Ayrıca, belirtilen ihtiyaçların doğru yorumlandıkları da sınanmalıdır.

6.2.4 Kabul Sınaması

Çalıştırılmadan önce sistemin son sınamasıdır.

Artık, yapay veriler yerine gerçek veriler kullanılır.

Bu sınama türü alfa sınaması veya beta sınaması olarak ta bilinir.

6.3 Doğrulama ve Geçerleme Yaşam Döngüsü

Gerçekleştirim aşamasına kadar olan süreçlerde doğrulama ve geçerleme işlemlerinin planlaması yapılır.

Planlama genellikle;
- alt-sistem,
- bütünleştirme,
- sistem ve
- kabul sınamalarının

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

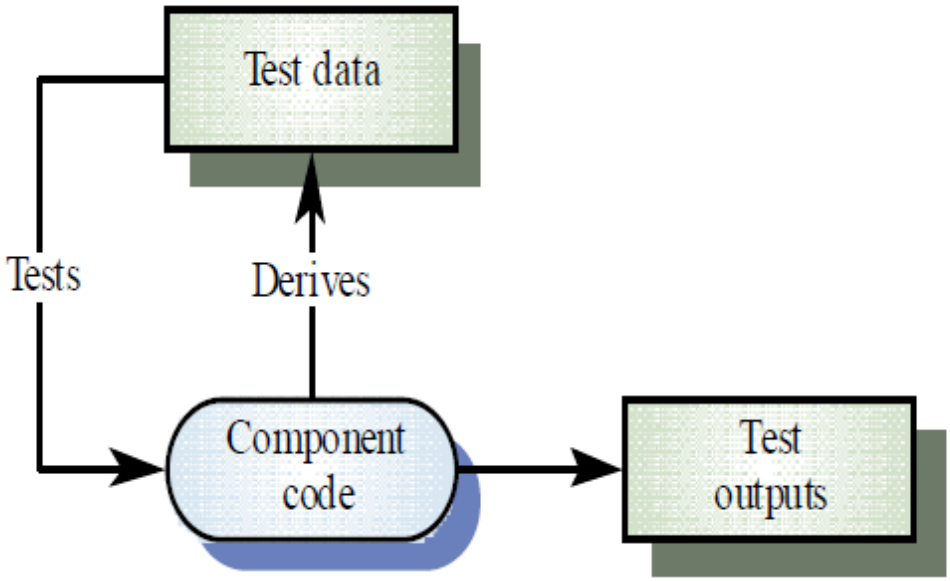
6.4 Sınama Yöntemleri

Her yazılım mühendisliği ürünü iki yoldan sınanır:

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

Beyaz Kutu Sınaması (White Box testing): İç işlemlerin belirtimlere uygun olarak yürütüldüğünün bileşenler tabanında sınanmasıdır.

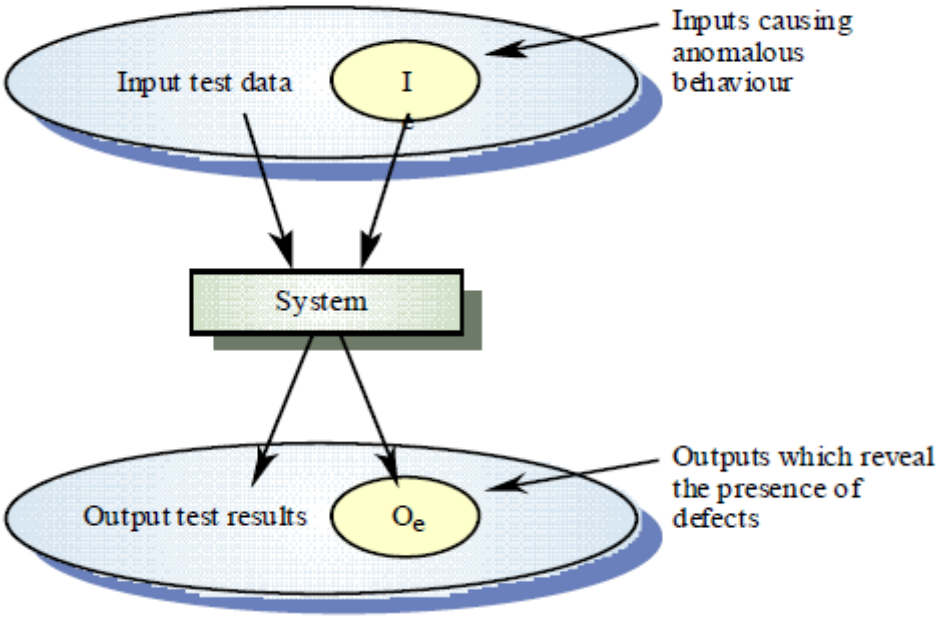
6.4.1 Beyaz Kutu Sınaması



Beyaz Kutu Sınaması

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

6.4.2 Kara Kutu Sınaması



Ürünlerin test edilmesi sırasında kullanılan en ilkel test metodudur. Bir takım test senaryolarının seçilip, yazılım kodundan bağımsız olarak takip edilmesi temeline dayanmaktadır. Bu yüzden ürünlerin fonksiyonel durumları ve inputlara verdikleri tepkilerin gözlenmesi uygulamada kullanılan kara kutu testlerinin kapsamını oluşturmaktadır. Bu noktada, yazılımın kodunda yapılan herhangi bir değişiklik veya data yapısındaki uyarlamalar kara kutu testleriyle kontrol edilen özellikler değildir.

Test edilecek olan uygulamanın kodu hiç dikkate alınmadan, sadece girdilerin ve çıktıların incelenmesi ile gerçekleştirilen test metodudur. 5 ayrı tekniği bilinir. Denklik sınıfı test tekniği, test verileri gruplanır. Gruplar içinde testler yapılır.

1. Uç nokta test tekniği, hataların genelde sınırlarda çıktığı varsayılarak sınır değerlerinde test yapılır.
2. Karar tablosu test tekniği, çok fazla test yapılması gereken uygulamalarda verilerin matrix haline getirilerek test edilmesi test edilmesidir.
3. Sistem durumu test tekniği, farklı durum geçişleri yer alan sistemlerin testleridir.
4. İş senaryosu test tekniği, use case dokümanlarının kullanıldığı test tekniğidir

6.5 Sınama ve Bütünleme Stratejileri

6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme

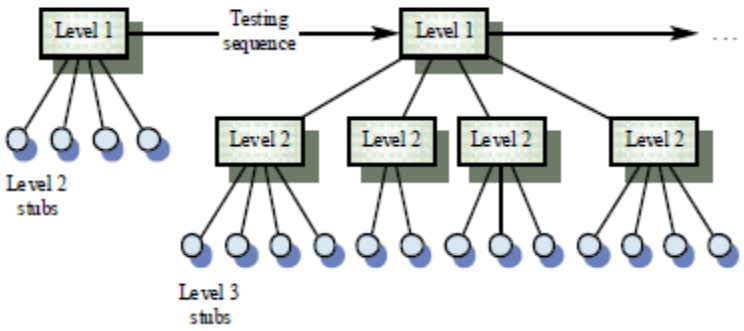
Yukarıdan-aşağıya bütünleştirmede önce sistemin üst düzeylerinin sınanması ve sonra aşağıya doğru olan düzeylere ilgili modülleri takılarak sınanması söz konusudur.

En üst noktadaki bileşen sınandıktan sonra alt düzeye geçilmelidir.

Alt bileşenler henüz hazırlanmamışlardır. Bu sebeple Koçanlar kullanılır. **Koçan**: Bir alt bileşenin, üst bileşen ile ara yüzünü temin eden, fakat işlevsel olarak hiçbir şey yapmayan çerçeve programlardır.

İki temel yaklaşım vardır:

- Düzey Öncelikli Bütünleştirme: En üst düzeyden başlanır ve aynı düzeydeki birimler bütünleştirilir.
- Derinlik Öncelikli Bütünleştirme: En üst düzeyden başlanır ve her dal soldan sağa olmak üzere ele alınır.



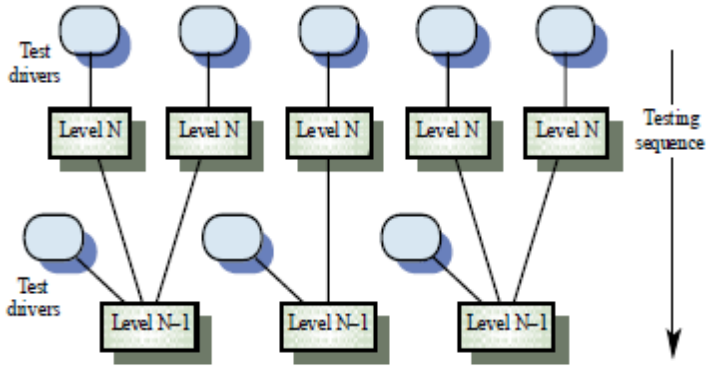
Yukarıdan Aşağıya Sınama

6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme

Önceki yöntemin tersine uygulama yapılır.

Önce en alt düzeydeki işçi birimler sınanır ve bir üst düzey ile sınanması gerektiğinde bu düzey bir sürücü ile temsil edilir.

Bu kez kodlama, bütünleştirme ve sınama, aşağı düzeylerden yukarı düzeylere doğru gelişir.



Aşağıdan Yukarıya Sınama

6.6 Sınama Planlaması

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

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 oluşturmasıdır. Sınama işlemi sonrasında bu belirtimlere,

- sınamayı yapan,
- sınama tarihi,
- bulunan hatalar ve açıklamaları türündeki bilgiler eklenerek sınama raporları oluşturulur.

6.8 Yaşam Döngüsü Boyunca Sınama Etkinlikleri

7. BAKIM

7.1 Giriş

Yazılımın dağıtılması ve kullanıma başlanmasından sonra yazılımda yapılacak değişiklikler yazılımın bakımı (software maintenance) olarak adlandırılır. Bu değişiklikler basit kodlama hatalarının düzeltilmesi (bug-fixes) şeklinde olabileceği gibi tasarımdan kaynaklanan hataların giderilmesi gibi daha kapsamlı değişiklikler şeklinde de olabilir. Yazılımın bakımı aslında yazılımın evrimleşmesidir. Yazılımın yaşamına devam edebilmesi için gerekli değişikliklerin uygulanmasıdır.

7.2 Kurulum

Kurulum teknik ekip tarafından restorana yapılacaktır.

7.3 Yerinde Destek Organizasyonu

Yerinde destek ekibi, kullanıcı alanında yerleşik olarak bulunan gerekli sayıda elemandan oluşan bir ekiptir.
Bu ekibin temel görevleri:

- Kullanıcıları ziyaret ederek sorunlarını belirlemeye çalışmak,
- Giderilebilen kullanıcı sorunlarını gidermek ve giderilemeyenleri üretim sahasındaki uygulama yazılımı destek ekibine iletmek,
- Kullanıcıya işbaşında uygulama eğitimi vermek,
- Kullanıcı sınama günlüklerini toplamak
- Yapılan tüm işlemleri konfigürasyon veri tabanına kaydetmek biçimindedir.

7.4 Yazılım Bakımı

Yazılım bakımı yapılırken şu programlardan faydalanılacaktır.

- Atlassian Jira Talep Takip (Issue Tracking) Yazılımı – Değişiklik taleplerinin girilerek yapılabilirlik analizinin başlatılması, değişiklik yönetim süreçlerinin izlenmesi ve proje ekibi üzerindeki görevlerin takibi için kullanılacak web tabanlı yazılım aracıdır.

- Atlassian Confluence (Wiki) Yazılımı – Değişiklik ve Yapılandırma yönetim süreçlerindeki tüm dokümantasyonların hazırlanması, saklanması ve erişilmesi için kullanılacak web tabanlı yazılım aracıdır.

- Atlassian Fisheye + Crucible Yazılımı – Kaynak kod deposu üzerinde gezinmek, kaynak kod dosyalarını görüntülemek ve sürümleri arasındaki değişiklikleri izlemek için Fisheye, proje ekibindeki yazılım geliştiricilerin yapacakları değişiklikleri gözden geçirmek, yorumlamak ve gerekirse yönlendirmek amacı ile Crucible yazılımı kullanılmaktadır. Her iki yazılım da web tabanlıdır.

8. SONUÇ

Sonuç olarak restoranlarda alternatif olabilecek derecede bir proje oluşturdum. Proje dokümantasyonunu yazılım mühendisliği kriterlerine uygun olarak yazmaya çalıştım. Projenin gereksinimlerini tanımladım.

9. KAYNAKLAR

http://www.bilgisite.com/yonetim/kaliteyonetim/yonetim_09.html

http://univera-ng.blogspot.com

http://www.bidb.itu.edu.tr/?d=1064

http://www.godoro.com/Divisions/Ehil/Mahzen/Java/J2EEEmpire/txt/html/document_SystemArchitectural.html

http://www.yarbis.yildiz.edu.tr/

http://tr.wikipedia.org/wiki/Ana_Sayfa

http://www.tutev.org.tr

http://taliphakanozturk.wordpress.com

http://www.yazilimprojesi.com

http://technet.microsoft.com

http://w3.gazi.edu.tr/~nyalcin/CASE.pdf

http://cse.cbu.edu.tr

ARİFOĞLU, Ali; Yazılım Mühendisliğine Giriş

KALIPSIZ, OYA; Yazılım Mühendisliği

ÇÖLKESEN, Rıfat, "Veri Yapıları ve Algoritmalar.

SARIDOĞAN, Dr. Erhan. ," Yazılım Mühendisliği Temelleri