Fırat Üniversitesi

15542516 Kemal Bayram Site Takip Programı

ÖNSÖZ

Teknolojinin de gelişmesiyle beraber yazılımda gelişimde baya yol kat etmiştir ve artık veri tabanları ortaya çıkmıştır ve veri tabanları sayesinde bir bilgi bilgisayar hafızasında düzenli olarak tutuluyor, görüntüleniyor, güncelleniyor ve silinme işlemi de yapılabiliyor.

Bir iş adamının yaptığı yapıların bilgilerinin veri tabanında tutulması ve bu verilere istediği anda ulaşmasına imkan sağlamaktadır.

Yaptığım bu projede iş adamının kasa durumlarının, bina durumlarının ve daha birçok iş adamını ilgilendiren bilgilerinin veri tabanında düzenli olarak tutulmasını bunların yanın da bu sistemin cevrim içi olarak paylaşılmasıyla(gerekli önlemler alındıktan sonra) işletme yöneticilerinin an an kasa durumunu satış durumunu takip edebilecek tir.

Elinizdeki bu belgede, yukarıda bahsi geçen projenin amacı, kapsamı, harekete geçirilmesiyle ilgili planlama, zaman – iş planlaması, ekip yapısının tanıtılması amaçlanmıştır.

İÇİNDEKİLER;

1)Giriş

Bu yazılım projesinde kişilerin yapıların takibinin sorunsuzca yönetilmesi sağlanmıştır. Sitelerimiz ve apartmanlarımız tam anlamıyla kontrol altına alan Site takip programımız, Sitelerin, Apartmanların, Dairelerin vs. kolaylıkla listelenmesine olanak sağlar ve her istediğiniz bilgiye rahatlıkla ulaşabilirsiniz.

Kolay kullanılabilirlik görsel anlamda zenginleştirilmiş ikonlar ve daha bir sürü yenilikçi fikirlerle bu projeyi hazırladık.

Genel Özellikler;

- Yeni siteleri kolaylıkla ekleyebilirsiniz.
- Günlük olarak kasa durumunu takip edebilirsiniz.
- Bir Apartmandaki Dairelerin durumunu kiralık veya satılık olarak ayarlayabilirsiniz.
- Ayarlamaya göre istenilen filtrede daireleri listeleyebilirsiniz.
- Günlük olarak satılan veya kiralanan daireleri listeleyebilirsiniz.
- Siteye veya Apartmana özel özellikler ekleyebilirsiniz.
- İstenilin dairenin satılık veya kiralık durumunda kolaylıkla güncellemeleri yapabilirsiniz.
- Ve daha birçok özellik…

1.1)Projenin temel amacı;

İş adamlarının yapılarını ve bu yapıların durumlarını tam anlamıyla takip edebilmeleri gelir gider durumunu ve daire durumunu takip edebilecekleri kolay kullanıma sahip görsel olarak kullanıcı canlısı program tasarlamak ve bu programla beraber iş adamlarının memnuniyetini kazanmaktır. Yapılan programda yanılmaz ve her zaman kesin sonuçlar vardır.

Programımız bilgisayar donanımlarımızı en etkin şekilde kullanır gereksiz yere kasmalar veya programın askıda kalması gibi olaylara şahit olmazsınız.

Projemiz bir ana giriş sayfası ve bu sayfa içinde Site ekleme, Site çıkarma, Apartman ekleme, Apartman çıkarma, kasa işlemleri gibi temel işlemler bulunacaktır. Ana formdan çeşitli yan formlara da ulaşılabilecek. Hassasiyetle tasarlanmış (ve bütün bilgilerin güvenlikle tutulduğu) veri tabanı olacaktır.

1.2)Tanımlamalar ve kısıtlamalar

Yönetici : Sistemin maddi durumunu kontrol edecek olan(iş adamı vs.).

Müşteri : Daireyi satın alacak veya kiralayacak kişi/kişiler.

Muhasebe: Müşteri, sisteme yeni yapılacak olan yapıları ekleyecek ve bu yapılardan satılan yada kiralananları belirtecek olan kişi

Temsilci: Yapılan binaları listeleyip onların durumuna göre gelen müşteri ile ilgilenip satışı yapacak olan kişi/kişiler.

2)Proje Planı

2.1)Giriş

Bu alanda yapılan projenin inceleme ve analizi yer almaktadır ve yapılan proje bu inceleme ve analizlere dayandırılmaktadır. İnceleme kısmında proje hakkında müşteriden en ince ayrıntısına kadar bilgi alınacaktır ve varsa bir önceki sistem tam anlamıyla incelenecektir. Bundan sonrada yapılabilirlik hesabı yapılıp eğer olumlu sonuç alınırsa projeye başlanacaktır.

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

Yanıtlar İçin Karşılıklar

0	Hiçbir etkisi yok .
1	Çok az etkisi var.
2	Etkisi var.
3	Ortalama etkisi var .
4	Önemli etkisi var.
5	Mutlaka olmalı, kaçınılmaz.

Ölçüm Parametresi	Sayı	Ağırlık	Çarpımı
Kullanıcı girdi sayısı	21	6	126
Kullanıcı çıktı sayısı	23	7	161
Kullanıcı sorgu sayısı	10	6	60
Kütük sayısı	11	15	165
Dışsal ara yüz sayısı	17	5	85
	Toplam sayı		597

2.2.1)İşlev noktası hesaplama

İN: İşlev noktası sayısı, AİN: Ayarlanmamış işlev noktası sayısı, TKF: Teknik karmaşıklık faktörü

İN = AİN x (0,65 x 0,01 x TKF)

İN = 597 x (0,65 x 0,01 x 41) = 159,1

2.2.2)Tahmini ulaşılacak satır sayısı

Satır Sayısı = İN x 30

Satır Sayısı = 4.773,105

2.2.3)Etkin maliyet modeli COCOMO

İş Gücü (K) = $a \times S^b$

Zaman (T) = $c \times K^d$

a,b,c,d : Her bir sistem için farklı olan katsayılardır.

S: Bin türünden satır sayısıdır.

İş Gücü (K) = $2,4 \times 4^{1,05}$

= 10,28 İş Gücü

Zaman (T) = $1,12 \times 10^{0,38}$

=1,68 Ay

2.3)Gantt Diyagramı

Kimlik	Görev Adı	Başlangıç	Bitiş	Süre
1	***Proje İsteklerinin Analizi***	**26.09.2017**	**04.10.2017**	**7g**
2	Projenin Tanımlanması	26.09.2017	27.09.2017	2g
3	Sistem Mimarisinin Belirlkenmesi	28.09.2017	29.09.2017	2g
4	İster Analizinin Hazırlanması	02.10.2017	04.10.2017	3g
5	***Proje Planı***	**05.10.2017**	**23.10.2017**	**13g**
6	Proje Zaman İş Planı	05.10.2017	09.10.2017	3g
7	Proje Ekip Yapısının Oluşturulması	09.10.2017	13.10.2017	5g
8	Planların Oluşturulması	16.10.2017	23.10.2017	6g
9	***Sistem Çözümleme***	**24.10.2017**	**21.11.2017**	**21g**
10	Mevcut Sistemin İncelenmesi	24.10.2017	24.10.2017	1g
11	Gereksenen Sistemin Mantıksal Modelleri	25.10.2017	01.11.2017	6g
12	Ara yüz Gerekleri	02.11.2017	10.11.2017	7g
13	Belgeleme Gerekleri	13.11.2017	21.11.2017	7g
14	***Sistem Tasarımı***	**22.11.2017**	**06.12.2017**	**11g**
15	Genel Tasarım	22.11.2017	24.11.2017	3g
16	Veri Tasarımı	27.11.2017	29.11.2017	3g
17	Süreç Tasarımı	30.11.2017	06.12.2017	5g
18	***Sistem Gerçekleştirimi***	**07.12.2017**	**13.12.2017**	**5g**
19	Ara yüz Gerçekleştirme	07.12.2017	08.12.2017	2g
20	Veri Tabanı Gerçekleştirme	08.12.2017	08.12.2017	1g
21	Kodlama	08.12.2017	13.12.2017	4g
22	***Doğrulama ve Gerçekleme***	13.12.2017	14.12.2017	2g

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.

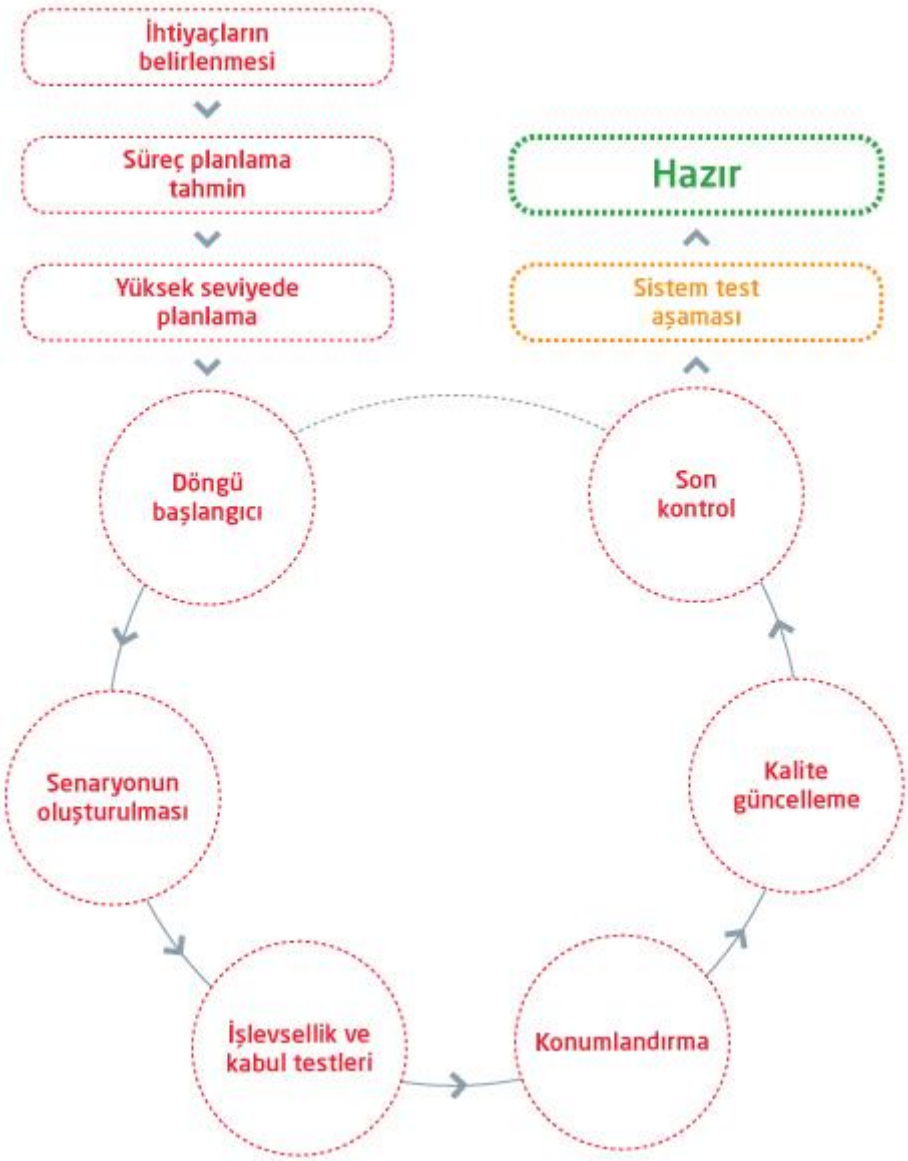
2.5)Önerilen Sistemin Teknik Tanımları

Sistemde, mobil uygulamalar geliştirilecek. Bu uygulamalar müşteri tarafından kullanılacak ve kullanılabilirlik açısından olabildiğince basit tasarlanacaklardır.
Yöneticinin işlemleri takibi yapabilmesini sağlayan ve denetlenebilen bir masaüstü program geliştirilecektir.
Bütün depoların bağlı olacağı temel bir veri tabanı oluşturulacaktır. Ayrı olarak depoların kendi içlerinde stok bilgilerini, personel bilgilerini vs. tutabileceği veri tabanları olacak ve bu ana veri tabanıyla iletişim halinde olacaktır.

2.6)Kullanılan Özel Geliştirme Araçları ve Ortamları

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 modelerinden biri olan Çevik(Agile) model kullanılmıştır.



Çevik(Agile) Model

Modern yazılım geliştirme konusunda 1990'lardaki çeşitli makalelerde bir şekilde karşımıza Agile Development (geliştirme modeli) çıkmaktadır. Ortaya atılan fikirler yeni olmasa da internetteki popüler arama motorlarının konu ile yakından ilgilenmesi, sürece hız kazandırmıştır. Buna bağlı olarak Agile Development atölye çalışmaları ise Jim Highsmith ve Bob Martin tarafından düzenlenmiştir. Başlangıçta amaç konuyla ilgili olan dağınık grupları bir araya getirmek ve çeşitli yazılım gruplarının konuya yaklaşımlarını anlamaktı. Robert Martin, sektörü bu tür teknikler çerçevesinde, tek bir çatı altında toplayacak bir manifesto oluşturulması konusunda istekli davrandı. Manifesto ve ortak çalışmalar sonunda ise Agile Development metodolojisini araştırmak ve geliştirmek üzere, kâr amacı gütmeyen bir oluşum doğdu.

Neden Agile Development?

Yazılımın güvenilirliğini artırmak, mühendislik süresini azaltmak ve uygulamaları belli bir sürede yetiştirmek gerektiğinde kurumlar, mühendislik maliyetini azaltan yollara ihtiyaç duymaktadır. Agile Development sürecinde olası hatalar minimuma indirilerek üretim maliyeti düşürülmektedir. Özellikle Agile programlama teknikleri, tamamen başarısız bir uygulamanın büyük risklerini ortadan kaldırmaktadır. Ancak uygulama dağıtılıp, sunuculara kurulduktan sonra bile güvenlik maliyetleri, uygulamadan gelecek kazancı düşürebilmektedir. Çoğu şirketin ulaşmak için çaba harcadığı güvenlik standartları, iyi tasarlanmış bir uygulamanın bakım sürecinde ancak müşterilerinin ihtiyaçlarına cevap vermedeki süreyi aşmamasıyla mümkündür. Agile Development Methodology bunu, geliştirme hatalarını minimuma indirerek ve sürekli sınama yaparak, hataların hızlıca bulunmasına olanak tanıyarak gerçekleştirir.

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.

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.

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 iyiye getirilir. 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.

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.

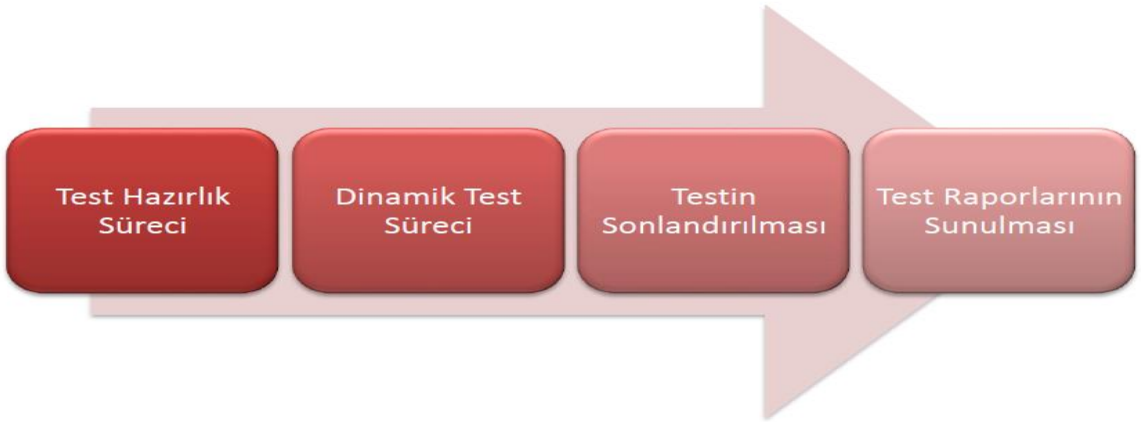
2.9)Eğitim Planı

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

Eğitim Planı

2.10)Test Planı

Çevik modelde döngü alanındaki işlevsellik ve kabul testi birde sistem teslim edilecek adımdan bir önceki adımda yapılan testlerdir.



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.
--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

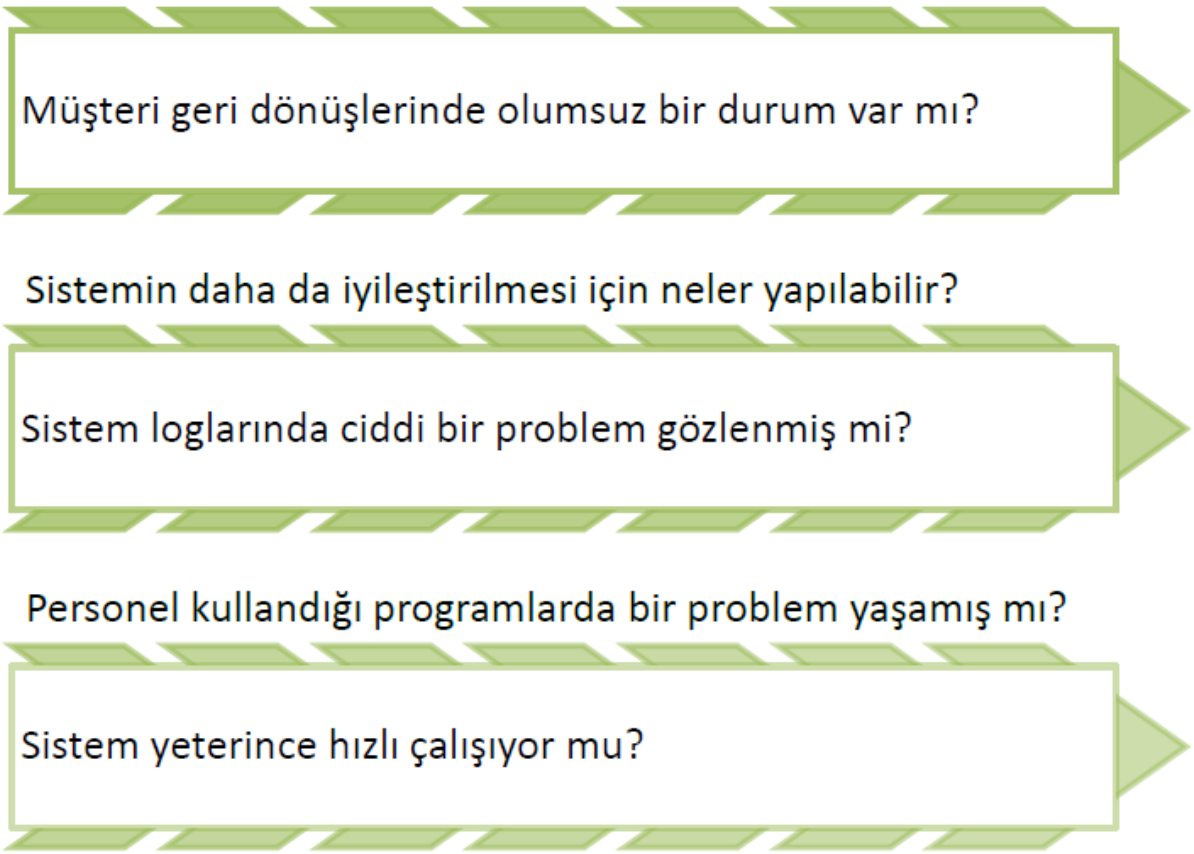
Bu süreç kodlama çalışmalarının bitmesine yakın bir dönemde başlayacaktır. Bulunan tüm hatalar çözülmeden ve testin sonlandırma kriterleri sağlanmadan sona ermez. Test edilecek yazılımın türüne göre, dinamik olarak uygulanacak test teknikleri ve bu tekniklerin uygulanma metotları farklılık gösterebilir.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



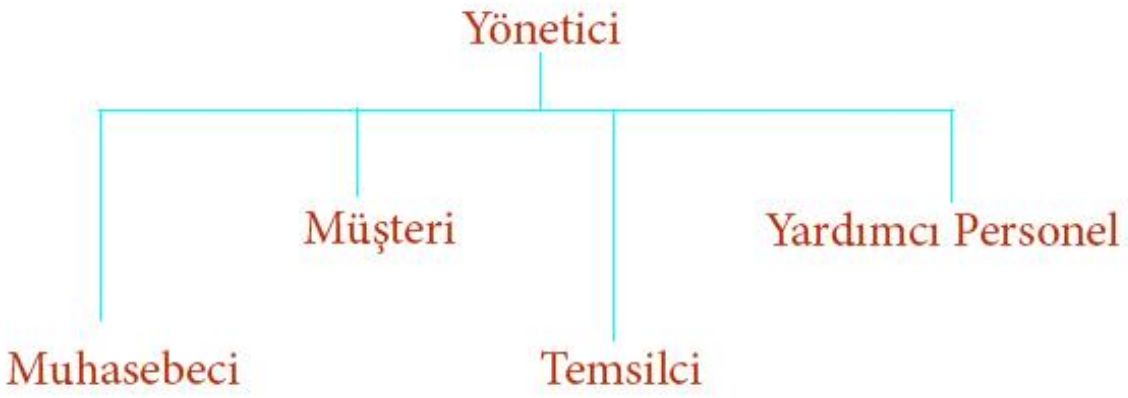
Aylık Bakım Planı

3)Sistem Çözümleme

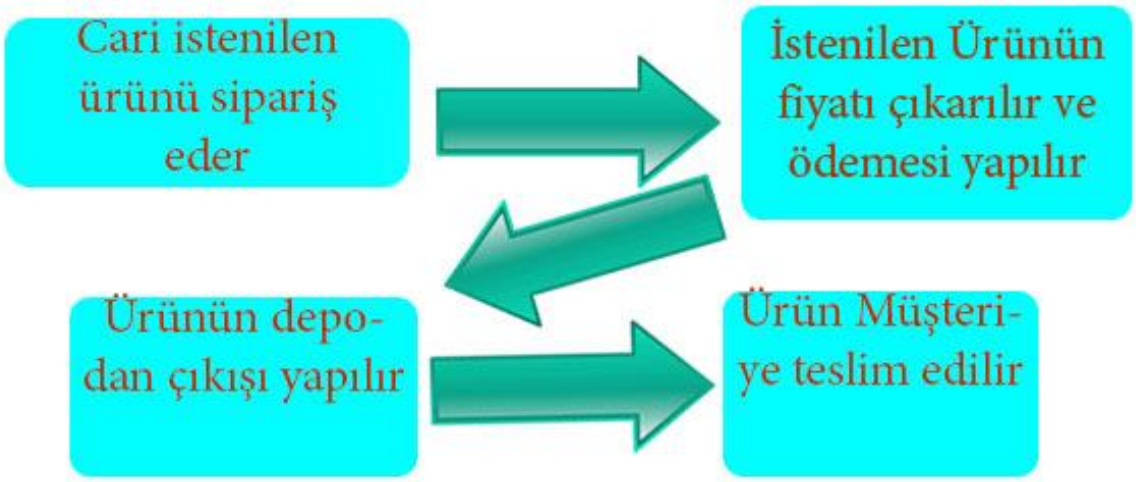
3.1)Mevcut Sistemin İncelenmesi

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

3.1.1)Örgüt Yapısı



3.1.2)İşlevsel Modeli



3.1.3)Var olan Yazılım/Donanım Kaynakları

Microsoft Project

MS Project, Microsoft tarafından geliştirilen ve satılan, proje yöneticilerine plan oluşturma, kaynakların görevlere atanması, aşama takibi, bütçe yönetimi ve iş yükü analizi gibi konularda yardımcı olması amacıyla tasarlanmış bir proje yönetimi yazılımıdır.
Bu projede Gantt Diyagramı çizilmesinde faydalanılmıştır.

Microsoft Word

Microsoft Word, dünyanın en popüler metin kontrol uygulamasıdır. Bu projede dokümantasyonun hazırlanmasında faydalanılmıştır.

ArgoUML

ArgoUML, ücretsiz olarak UML diyagramlarını çizmeye, modellemeye yarayan kullanımı kolay bir UML çizim programıdır. Bu projede diyagramların çiziminde faydalanılmıştır.

Adobe Photoshop CC

Photoshop, piksel tabanlı görüntü, resim ve fotoğraf düzenlemede bir tek biçim olan, Adobe Sistem'in sayısal fotoğraf işleme yazılımıdır.
Bu projede belirli resimlerin çizilmesinde faydalanılmıştır.

Microsoft Visio

Gantt diyagramı, kapsam diyagramı ve örgüt yapısı çıkarılmasında kullanılmıştır.

SmartDraw

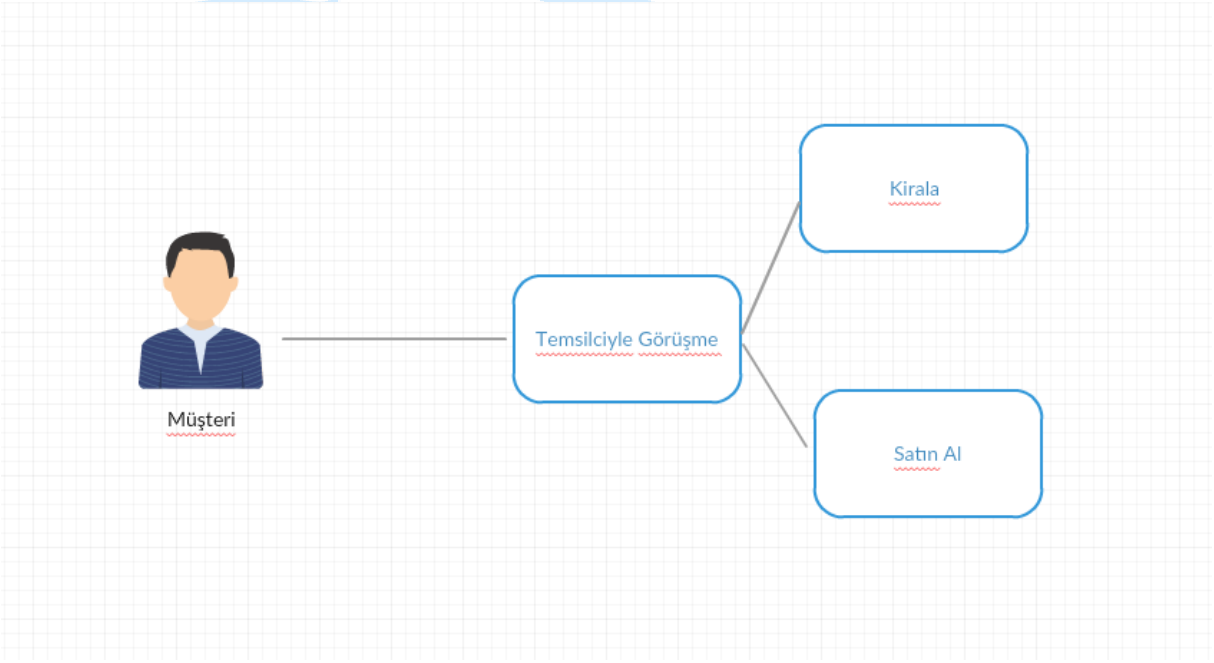
Use Case Diyagramları hazırlanırken faydalanılmıştır

3.2)Gereksenen Sistemin Mantıksal Modeli

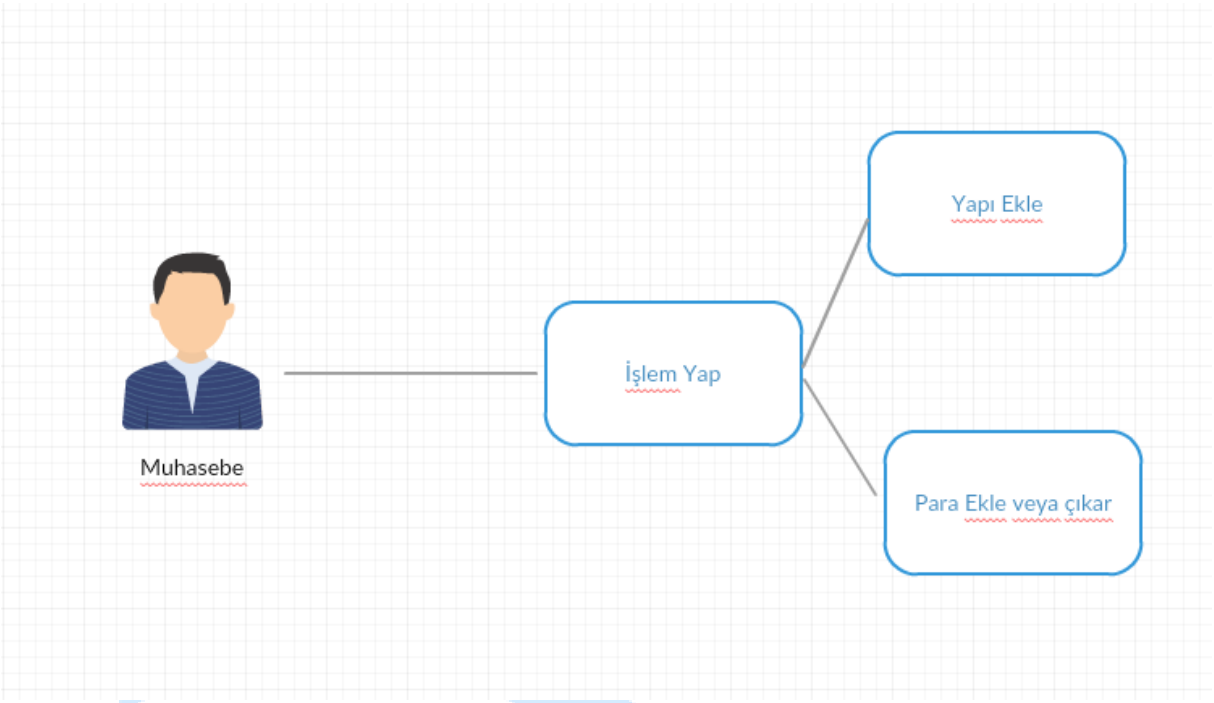
3.2.1)Giriş

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.

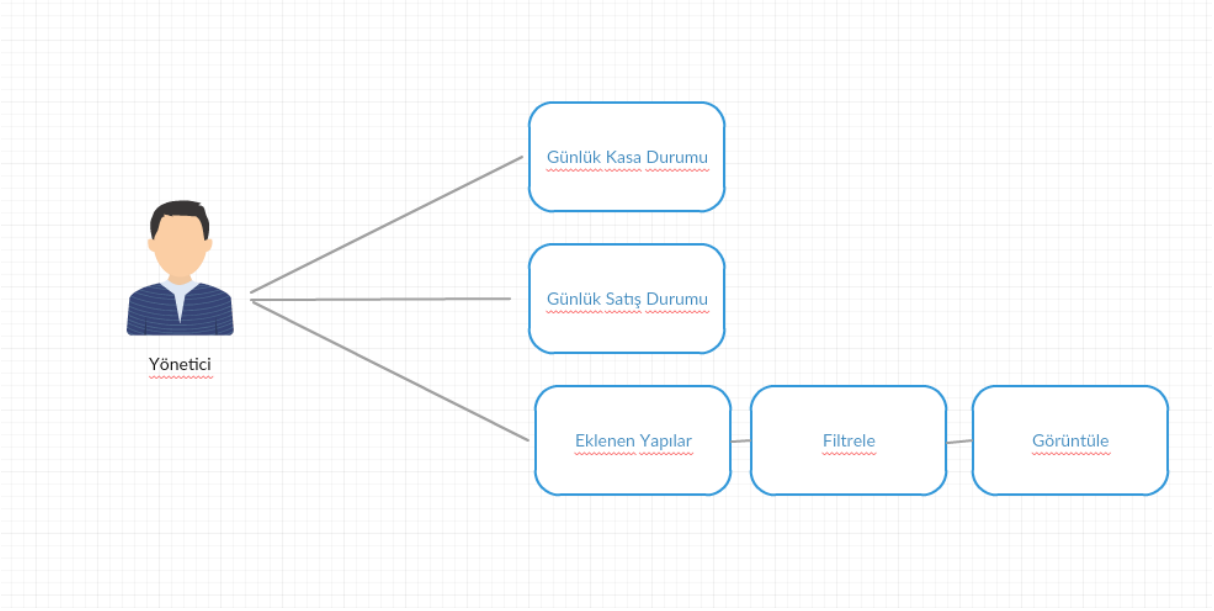
3.2.3)Use Case Diyagramları



Use Case Diyagramı(Müşteri)

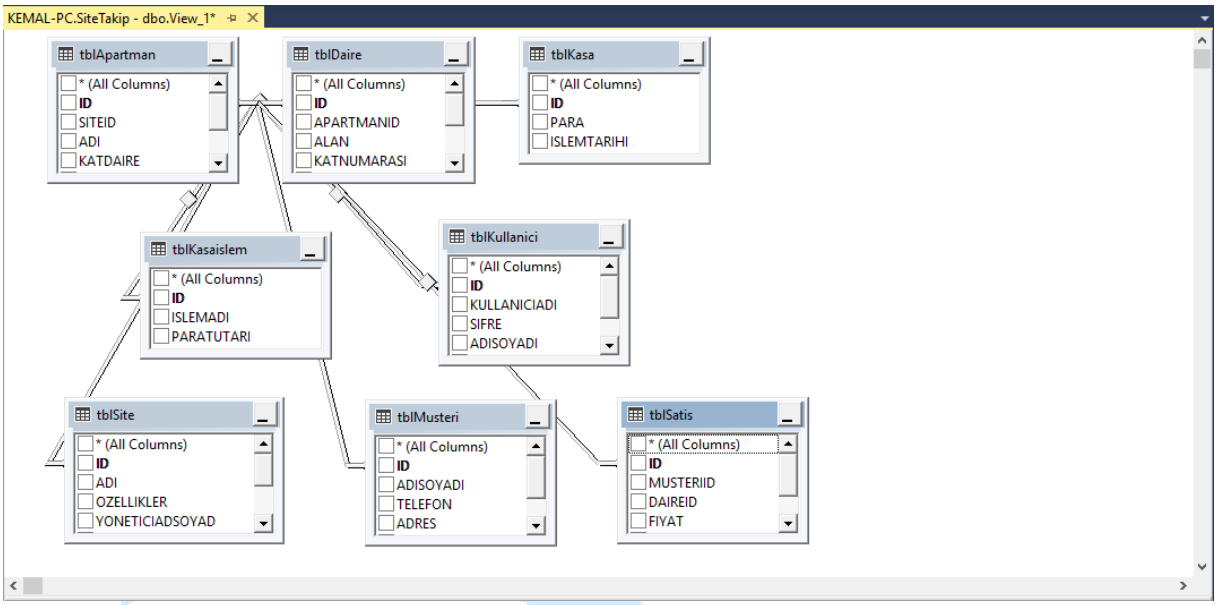


Use Case Diyagramı(Muhasebe)



Use Case Diyagramı(Yönetici)

3.2.4)Bilgi Sistemleri/Nesneler



Sınıf Diyagramı

3.2.6)Veri Sözlüğü

KEMAL-PC.SiteTakip - dbo.tblApartman

Column Name	Data Type	Allow Nulls
ID	int	☐
SITEID	int	☑
KATDAIRE	int	☑
ACIKLAMA	text	☑
OZELLIKLER	text	☑
KATSAYISI	int	☑
		☐

Apartman

KEMAL-PC.SiteTakip - dbo.tblDaire | KEMAL-PC.SiteTakip - dbo.tbl

Column Name	Data Type	Allow Nulls
ID	int	☐
APARTMANID	int	☑
ALAN	int	☑
KATNUMARASI	int	☑
TIPI	nchar(10)	☑
FIYAT	numeric(18, 2)	☐
MUSTERINO	int	☑
		☐

Daire

KEMAL-PC.SiteTakip - dbo.tblKasa | KEMAL-PC.SiteTakip - dbo.tblD

Column Name	Data Type	Allow Nulls
ID	int	☐
PARA	decimal(18, 2)	☑
ISLEMTARIHI	datetime	☑
		☐

Kasa

KEMAL-PC.SiteTakip - dbo.tblMusteri | KEMAL-PC.SiteTakip - dbo.

Column Name	Data Type	Allow Nulls
ID	int	☐
ADISOYADI	nchar(50)	☑
TELEFON	nchar(11)	☑
ADRES	nchar(70)	☑
TARIH	datetime	☑
		☐

Musteri

KEMAL-PC.SiteTakip - dbo.tblSatis | KEMAL-PC.SiteTakip - dbo.tbl

Column Name	Data Type	Allow Nulls
ID	int	☐
MUSTERIID	int	☑
DAIREID	int	☑
FIYAT	decimal(18, 2)	☑
TIPI	nchar(10)	☑
TARIH	datetime	☑
		☐

Satis

KEMAL-PC.SiteTakip - dbo.tblSite | KEMAL-PC.SiteTakip - dbo.tblS

Column Name	Data Type	Allow Nulls
ID	int	☐
ADI	nchar(40)	☑
OZELLİKLER	text	☑
YONETICIADSOYAD	nchar(70)	☑
YONETİCİTELEFON	nchar(11)	☑
SITETELEFON	nchar(11)	☑
SITEADRES	nchar(50)	☑
ACIKLAMA	text	☑
		☐

Site

KEMAL-PC.SiteTakip - dbo.tblKasaislem | KEMAL-PC.SiteTa

	ID	ISLEMADI	PARATUTARI
▶	2	Satılık ...	450000,00
	3	Para Cekme ...	100,00
	4	Kiralık ...	750,00
	5	Para Yatırma ...	1500,00
	6	Satılık ...	450000,00
	7	Para Cekme ...	1000000,00
*	NULL	NULL	NULL

Kasa İşlem

KEMAL-PC.SiteTakip - dbo.tblKasaislem | KEMAL-PC.SiteTakip - dbo.tblKullanici

	Column Name	Data Type	Allow Nulls
🔑	ID	int	☐
	KULLANICIADI	nchar(15)	☑
	SIFRE	nchar(20)	☑
	ADISOYADI	nchar(50)	☑
	TELEFON	nchar(14)	☑
	EPOSTA	nchar(50)	☑
			☐

Kullanıcı İşlem

3.2.7)Başarım Gerekleri

- Sistem hızlı çalışmalı
- Ara yüz basit olmalı
- Personel ekip yapısı düzgün seçilmeli
- Güvenlik sorunları çıkmamalı
- Kullanıcılardan geri dönüş olmalı

3.4)Belgeleme Gerekleri

3.4.1)Geliştirme Sürecinin Belgelenmesi

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

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

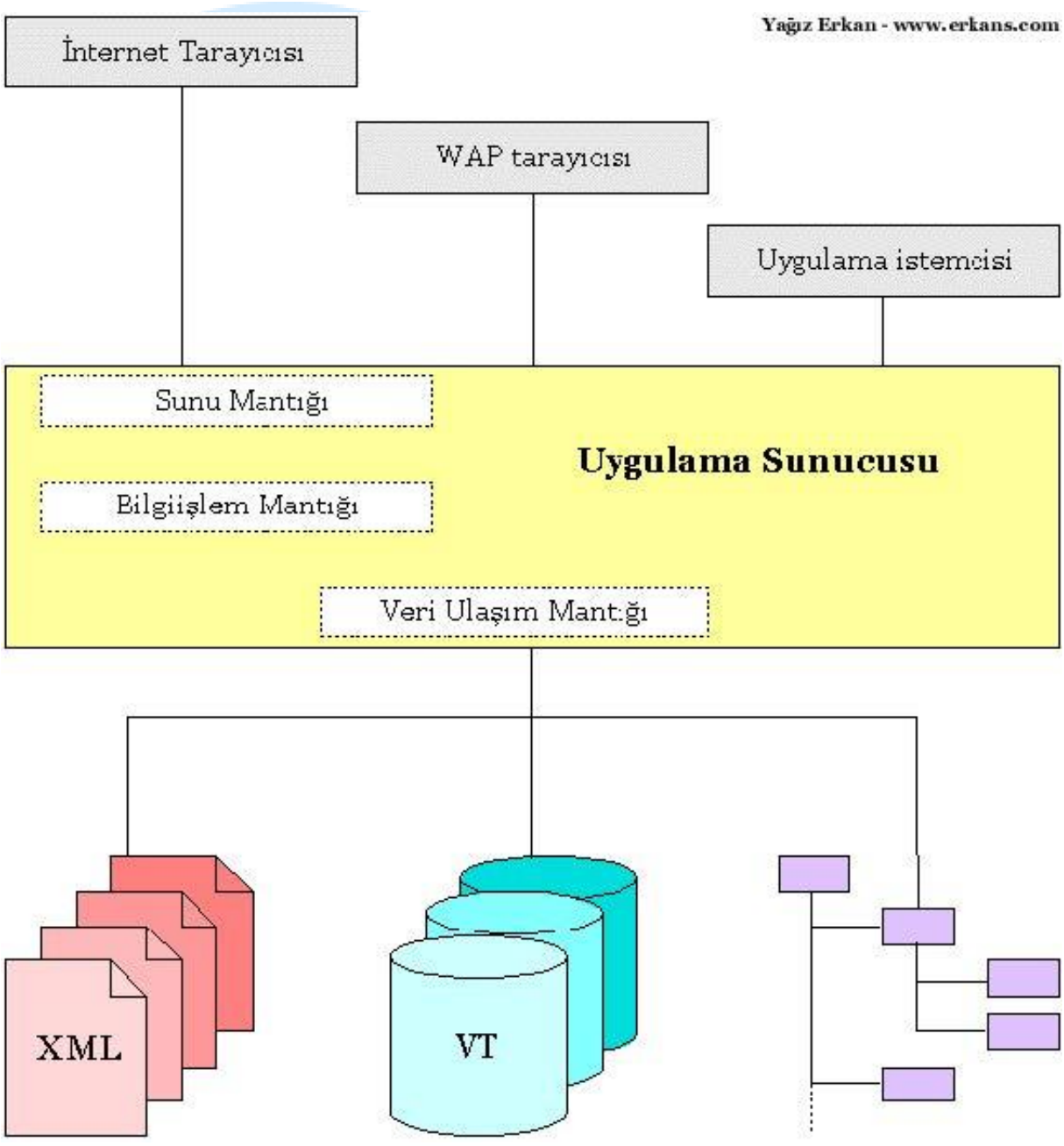
3.4.2)Eğitim Belgeleri

Çalışan personele sistemin bir parçası olabilmesi için eğitim verilmesi gerekmektedir. Bu eğitimin içeriği; sistemin işleyişi, personelin sistemdeki yeri, kullanacağı program hakkında bilgi verilmesi olacaktır.

3.4.3)Kullanıcı El Kitapları

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

4) SİSTEM TASARIMI



4.1.1)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.
Uygulama ya da İş Katı (Application/Business Tier): Uygulamanın hedef aldığı ve gereklerini tatmin etmek için geliştirildiği işe dayalı tüm bilgiişlem bu katta toplanır.
Entegrasyon Katı (Integration Tier): Bu kat, uygulamanın görevini yerine getirmesi için gerekli olan sistem dışı yazılımlara, sistemlere ya da veri tabanlarına bağlantıları sağlamakla yükümlüdür.
Kaynak Katı (Resource Tier): Bilgiişlem için gerekli veriler ve dış servisler bu katı oluşturur.

4.1.2)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)Veri Tasarımı

4.2.1)Tablo Tanımları

Ürünler: Ürün özelliklerinin tutulacağı tablodur.
Kullanıcı: Kullanıcı bilgilerinin tutulacağı tablodur.
Stok: Ürün stok bilgilerinin tutulacağı tablodur.
Fiyat: Ürün fiyatları ve tiplerinin tutulacağı tablodur.
Bakiye: Üyelerin bakiyelerinin tutulacağı tablodur.
Not: Üyenin oluşturduğu notların tutulacağı tablodur.
Kampanya: Kampanyalı ürünlerin tutulacağı tablodur.
İstatistik: İstatistiki bilgilerin tutulacağı tablodur.

4.3)Süreç Tasarımı

4.3.1)Genel Tasarım

Hedef, yapı diyagramı çizmektir. Veri akış diyagramından yola çıkılarak işlem türlerinin bölgeleri tanımlanır ve bu bölgelere karşı düşecek yapısal öğeler ortaya çıkarılmış olur. İstenilen yapı diyagramı kontrol hiyerarşisini de göstermektedir.

4.3.2)Modüller

Kullanıcı Modülü: Her market müşterisinin erişebileceği ve kullanabileceği modül
Personel Modülü: Market personelinin paketleri hazırlamasında kullanabileceği modül
Kasiyer Modülü: Sipariş tamamlama işlemlerinin kasiyer tarafından yapılabileceği modül
Yönetici Modülü: Genel market yönetiminin görülebileceği yöneticiler tarafından kullanılabilecek olan modül

4.3.2.1)Kullanıcı Arabirimi

Sistemde 4 adet kullanıcı profili olacaktır.
Market Müşterisi: Müşterilerin alışveriş, not tutma, üyelik alma gibi işlemlerini yapabilecektir.
Kasiyer: Kullanıcıların alışveriş bitiminde yapacağı işlemleri yürüteceği kişidir.
Personel: Kullanıcı alışveriş yaptıkça arka planda sepetleri hazırlayacak kişidir.
Yönetici: Mağazalardaki genel durumun görülebileceği akıllı listelere ulaşılabilecek kişidir.

4.3.2.2)Entegrasyon ve Test Gereksinimleri

Sistemde 4 farklı program geliştirilecektir. Hepsinin ayrı ayrı ve birbirleriyle ilişkili şekilde test edilmesi gerekmektedir. Kullanıcı modülü için ayrı ayrı akıllı telefonlarda denenmelidir. Diğer programların tam verimli çalışıp çalışmadığı test ekibi tarafından detaylı bir şekilde test edilmelidir. Bütün modüller test edilip çalıştığına karar verildiğinde hepsinin veri tabanıyla kullanılarak test işlemi yapılmalıdır.

4.4)Ortak Alt Sistemlerin Tasarımı

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

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.

Güvenlik Alt Sistemi

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

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.

Veri İletişim Alt Sistemi

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)

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.

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;

1. C#

Bu dilleri kısaca açıklamak gerekirse ;

ECMA tarafından C# dilinin tasarım hedefleri şöyle sıralanır:

- *C# basit, modern, genel-amaçlı, nesneye yönelik programlama dili olarak tasarlanmıştır.*
- *Çünkü yazılımın sağlamlılığı, güvenirliliği ve programcıların üretkenliliği önemlidir. C# yazılım dili, güçlü tipleme kontrolü (strong type checking), dizin sınırlar kontrolü (array bounds checking), tanımlanmamış değişkenlerin kullanım tespiti, (source code portability), ve otomatik artık veri toplama gibi özelliklerine sahiptir.*
- *Programcı portatifliği özellikle C ve C++ dilleri ile tecrübesi olanlar için çok önemlidir.*
- *Enternasyonal hale koymak için verilen destek çok önemlidir.*
- *C# programlama dili sunucu ve gömülü sistemler için tasarlanmıştır. Bununla birlikte C# programlama dili en basit işlevselli fonksiyondan işletim sistemini kullanan en teferruatlısına kadar kapsamaktadır.*
- *C# uygulamaları hafıza ve işlemci gereksinimleri ile tutumlu olmak uzere tasarlanmıştır. Buna rağmen C# programlama dili performans açısından C veya assembly dili ile rekabet etmek için tasarlanmamıştır.*

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)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)

5.2.2.3)Veri Tabanı Dilleri ve Arabirimleri

Bu projede Oracle VTYS olduğundan PL/SQL dili kullanılacaktır. **PL/SQL,** (**Procedural Language/Structured Query Language**), Oracle tarafından geliştirilen Oracle veritabanı sistemlerine özel dildir. Oracle veri tabanı sistemlerinde tetikleyici(trigger) ve Saklı yordam (stored procedure) yazmak üzere geliştirilmiş temel sql komutlarının yanında programlamada akış kontrollerini ve değişkenleri kullanmaya olanak sağlayan yani yapısal dillere ait özelliklerin standart SQL'e eklenmesi sonucu oluşan bir dildir. Ada dili örnek alınarak tasarlanmıştır.

5.2.2.4)Veri Tabanı Sistem Ortamı

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

5.2.2.5)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 gereksinim-lerini 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.

5.2.2.6)CASE Araç ve Ortamları

Microsoft Project: Gantt Diyagramı çizilmesinde faydalanılmıştır.

Microsoft Word: Dokümantasyonun hazırlanmasında faydalanılmıştır.

ArgoUML: Diyagramların çiziminde faydalanılmıştır.

Adobe Photoshop CC: belirli resimlerin çizilmesinde, düzenlenmesinde faydalanılmıştır.

Microsoft Visio: Kapsam diyagramı ve örgüt yapısı çiziminde yararlanılmıştır.

Smart Draw: Use Case Diyagramları hazırlanırken faydalanılmıştır.

*5.2.2.7)Kodlama Stili

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.

Kod Biçimlemesi

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

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.

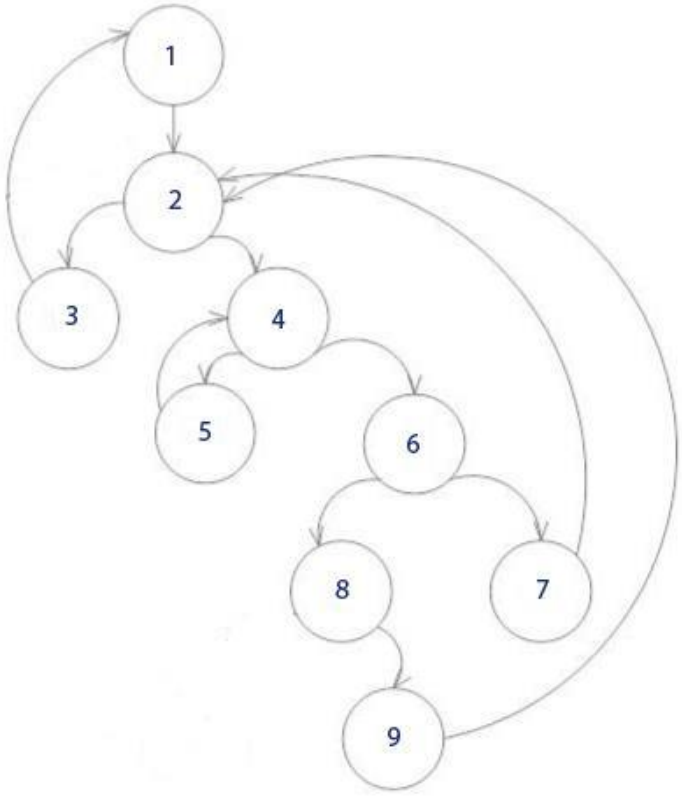
Yapısal Programlama Yapıları

Program kodlarının, okunabilirlik, anlaşılabilirlik, bakım kolaylığı gibi kalite etmenlerinin sağlanması ve perogram 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

5.2.2.8)Program Karmaşıklığı

Programın Çizgi Biçimine Dönüştürülmesi



McCabe Çizim Biçimi

McCabe Karmaşıklık Ölçütü Hesaplama

k = 12 Kenar sayısı

d = 9 Düğüm sayısı

p = 1 Bileşen sayısı

$V(G)= k – d + 2p$

$V(G)=12 - 9 + 2 \times 1 = 5$

5.2.2.9)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.

5.2.2.10)Kod Gözden Geçirme

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

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.

5.2.3)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?

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?

5.2.4)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?

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?

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?

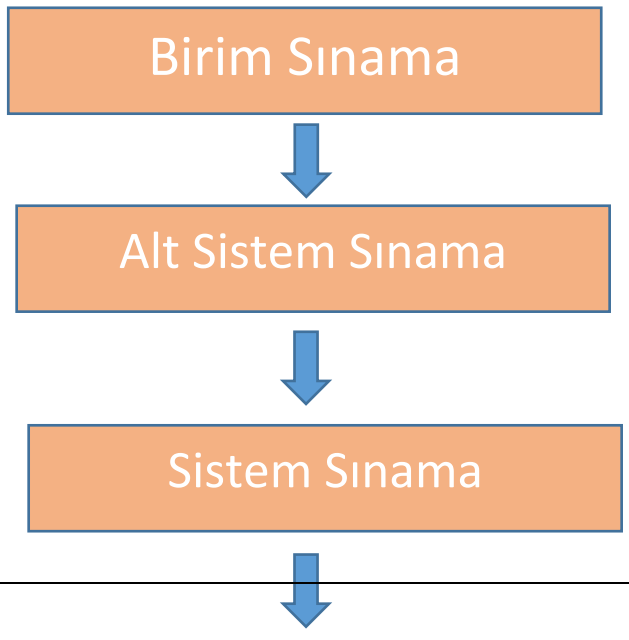
6) DOĞRULAMA VE GEÇERLEME

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:



Birim Sınama

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

Alt-Sistem Sınama

Alt-sistemler modüllerin bütünleştirilmeleri ile ortaya çıkarlar.
Yine bağımsız olarak sınamaları yapılmalıdır.
Bu aşamada en çok hata arayüzlerde bulunmaktadır. Bu yüzden arayüz hatalarına doğru yoğunlaşılmalıdır.

Sistem Sınaması

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

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.

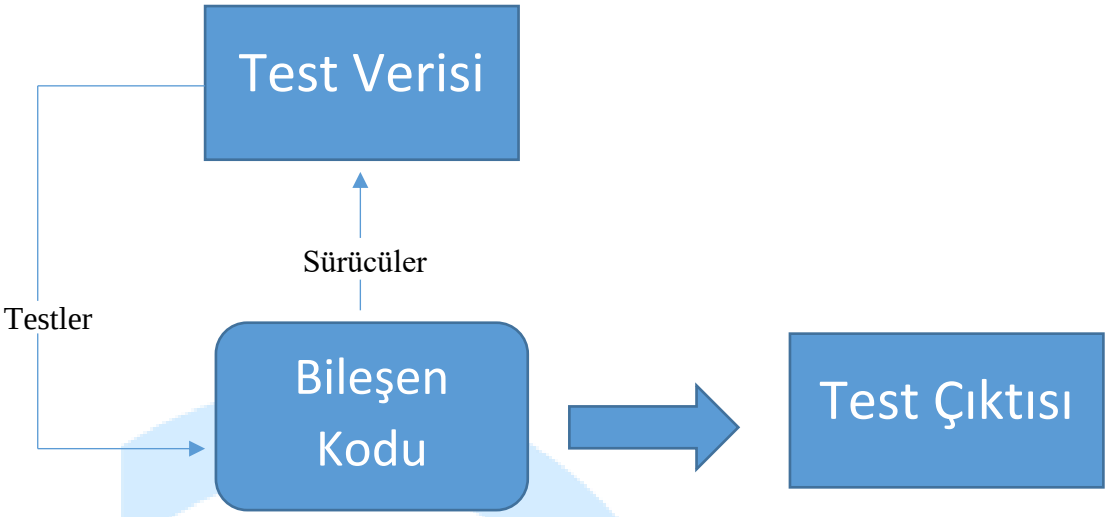
6.3)Doğrulama ve Geçerleme Yaşam Döngüsü

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

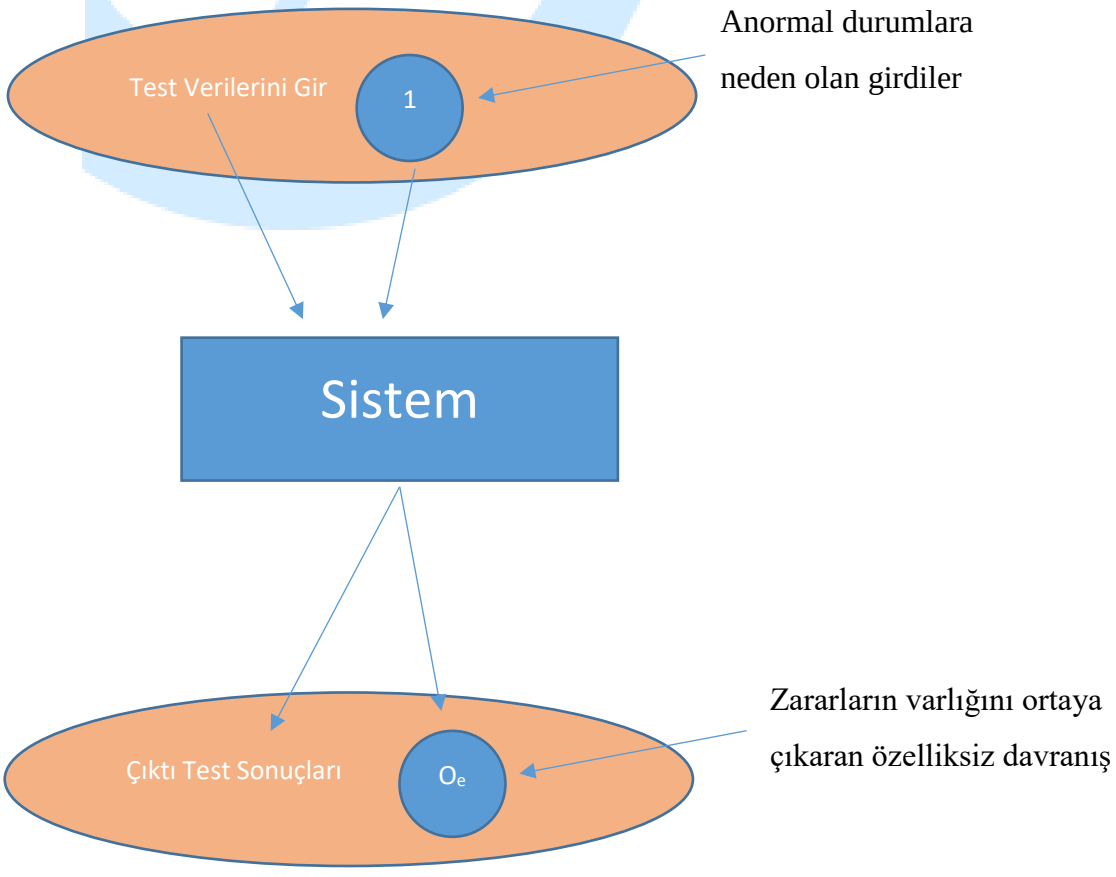
Her yazılım Mühendisliği ürünü iki yoldan sınanır:
Kara Kutu Sınaması (Black-Box testing): Sistemin tümüne yönelik işlevlerin doğru yürütüldüğünün testidir. Sistem şartnamesinin gerekleri incelenir.
Beyaz Kutu Sınaması (White Box testing): İç işlemlerin belirtimlere uygun olarak yürütüldüğünün bileşenler tabanında sınanmasıdır.

Beyaz Kutu Sınaması



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

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

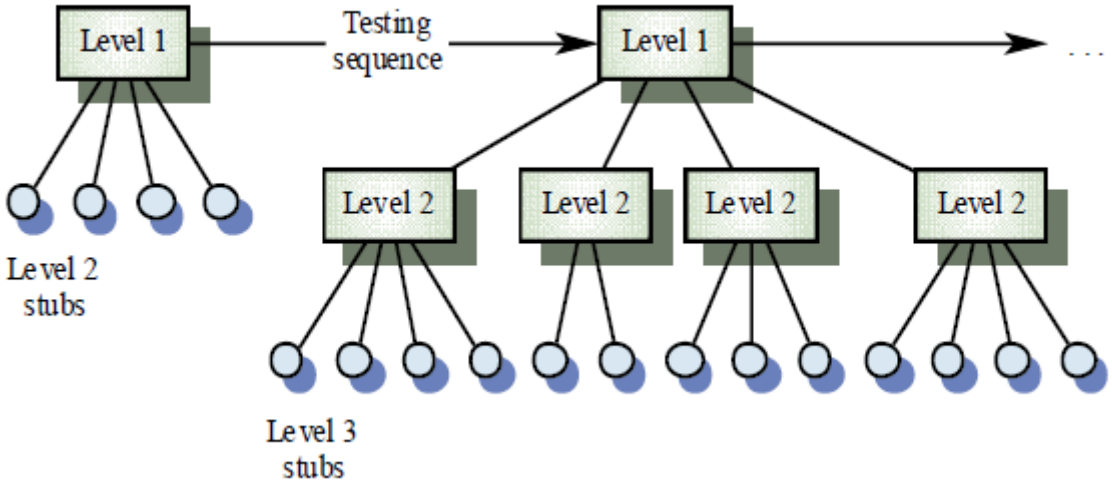
6.5)Sınama ve Bütünleme Stratejileri

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.

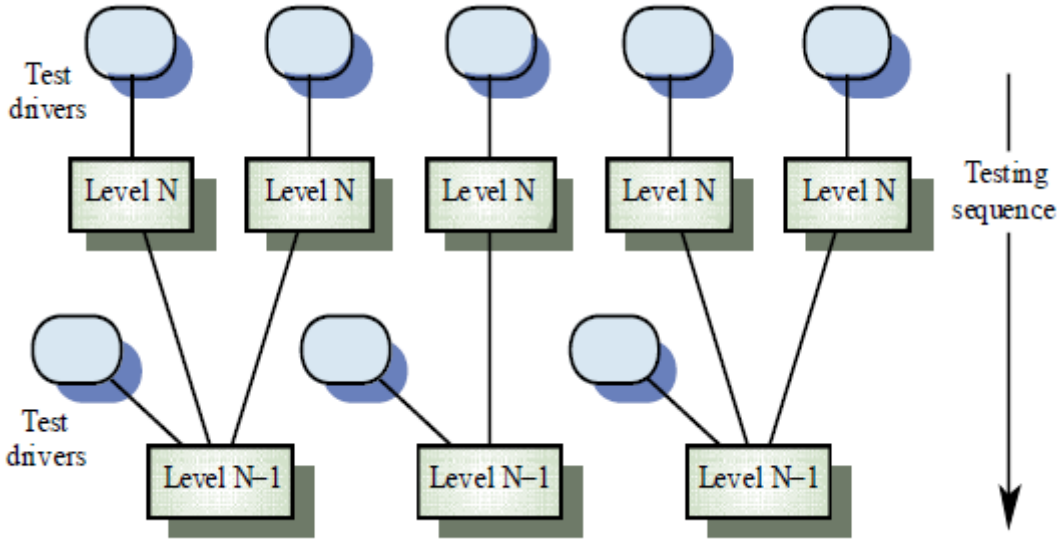
- Derinlik Öncelikli Bütünleştirme: En üst düzeyden başlanır ve her dal soldan sağa olmak üzere ele alınır.



Yukarıdan Aşağıya Sınama

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 Yukarı 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.

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 marketlere yapılacaktır. Müşterilerin kullanacağı uygulamalar ise Google Play veya App Store'dan ücretsiz olarak indirilip kolayca kurulabilecektir.

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

Yapılan bu program sayesinde yöneticiler artık depolarındaki malları çok rahatlıkla her yerden takip edebilecek istenilen kasalara veya bilgisayara yetki göndermesi yapabilecekler mobil telefon uygulaması sayesinde. Ve bu uygulamayı buradaki programları ve programlama dillerini yeterince öğrenince bu yazılım projesini gerçekten geliştirmeyi ve geliştirdiğim bu yazılımı da satmayı planlıyorum…

9)Kaynaklar

http://e-bergi.com/y/Cevik-Modelleme-ve-Cevik-Yazilim-Gelistirme

http://www.innova.com.tr/uygulama-gelistirme-yasam-dongusu.asp#Tab3-tab

http://docplayer.biz.tr/820320-Dinamo-bakim-plansiz-periyodik-onleyici-kestirimci-fabrika-bakim-planlama.html

https://tr.wikipedia.org/wiki/C_Sharp

https://tr.wikipedia.org/wiki/%C4%B0OS

https://tr.wikipedia.org/wiki/Android_(i%C5%9Fletim_sistemi)

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

http://univera-ng.blogspot.com

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

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