# Hafta -1

Dönem sonu raporları IEEE SRS (IEEE Software requirements spesification) Formatında olacak. **Yazılım Yaşam Döngüsü: Planlama-Çözümleme-Tasarım-Gerçekleştirme-Bakım. (P.Ç.T.G.B.)** *Yazılım Geliştirme süreci;*

* Gereksinim analizi (requirements analysis)
* Spesifikasyon
* Mimari (Architecture)
* Dizayn-Tasarım (Design)
* İmplementasyon (Programlama)
* Test
* Sahaya yerleştirme (Kurulum yada dağıtımda denilebilir)
* Bakım ve Onarım (Maintenance) User Story (Kullanıcı hikayeleri) demek.

**Yazılım gereksinimleri tanım belgesinde (YGTB);** geliştirilecek olan yazılımın işlevsel ve işlevsel olmayan gereksinimleri (ihtiyaçları) tanımlanır. Bunlar Use-Case, Kullanıcı arayüzleri, Kısıtlamalar, User Storyler, Sistemin analizleri, Sistemin tasarımı, ER diyagramlarıdır.

# PROJENİN AMAÇ KISMI:

Sorun analizi ve hedef analizi bu bölümde yazılır. Projenin kapsamını, tanımlarını, kısaltmalarını ve referansları (kaynakları) yine bu bölümde yazılır.

**Proje dökümanları;** Times New Roman 12 punto 1,5 satır aralığı ve iki yana yaslı şekilde olmalı.

*Bir projede bulunması gereken bölümler ve açıklamaları:*

1. bölüm- Tanıtım: Projenin amacı, kapsamı (hangi soruna çözüm bulduğu) ve tanıtımı yapılır.
2. bölüm- Planlama: Yazılım yaşam döngüsünün ilk fazı, planlamadır. Bu kısımda iş-zaman diyagramı olan GANT diyagramı çizilir. Ekip yapısı şematik olarak verilir. Bu bölümde insan kaynaklarının, donanım kaynaklarının ve yazılım kaynaklarının planlanması yapılır.
3. bölüm- Çözümleme: Bu bölümde mevcut sistemin analizi, incelenmesi ve sistemin yetemediği, eksik kalan noktalarını nasıl tamamlayacağımızın belirtilmesi gerekiyor. Use case diyagramları kullanılıyor.

İhtiyaçları belirlerken arayüzler, kısıtlar, use-case ve user storyleri kullanarak belirliyoruz. Arayüzler bu noktada önemli.

1. bölüm- Tasarım: Sistemin dizaynı, mantıksal tasarımı bu bölümde yapılır. Sistem içinde kullandığımız veritabanları, veri akışları, aktivity diyagramları ile sistemin arka planında ne olduğunu tasarlanır.
2. bölüm- Gerçekleştirme: Önerilen sistem gerçekleştirilirken hangi programlama dili kullanıldı, hangi araçların hangi teknolojilerin neden seçildiği bu bölümde açıklanır. Veritabanı sisteminde hangi mimari kullanıldığı bu bölümde açıklanır.
3. bölüm- Test: Yazılım yaşam döngüsünün her fazının doğal bir aktivitesi olarak tanımlanmıştır. Bu bölümde test planı verilir yani doğrulama ve geçerleme işlemlerinin nasıl yapılacağına ilişkin gant diyagramı olarak verilir.
4. bölüm- Bakım: Yazılım kurulum aktiviteleri ve sonrasında nasıl destek verileceği tarif edilmelidir. 8.bölüm- Sonuç: Projede neler yapıldığı, neler kazanıldığı eksik kalan kısımların özet geçildiği bölüm. 9.bölüm- Kaynaklar: Proje sürecinde kullanılan kaynaklar numaralandırılarak yazılır. Hangi sırada olduğu, nerede kullanıldığı, hangi kaynaktan alındığı yazılır.

# PROJE YÖNETİMİ

Proje nedir: Belirli bir yerde (belirli bir sahada) belirli bir zaman ve bütçe çerçevesinde, belirli bir mali doğrultuda, bir başlama ve bitiş noktası olan, önceden belirlenmiş amaçların ulaşılmasını sağlayacak olan faaliyetler topluluğudur.

Proje neleri sağlamalıdır:

* + Belirli bir yerde uygulanmalı yani uygulama alanı olmalı.
  + Belirli bir takvimi zaman aralığı olmalı
  + Bütçesi olmalı
  + Ve tanımlanmış faaliyetleri olmalıdır.

**Her fikir bir proje değildir**. Bir fikrin proje olabilmesi için, şu sorulara cevap verebilmeli.

* + Nerede uygulayacaksınız.
  + Ne kadar sürede bitireceksiniz
  + Belirli bir yer ve sürede gerçekleştireceğiniz faaliyetler için ne kadar mali kaynağa ihtiyaç var.
  + Bu sürede ve yerde hangi faaliyetleri gerçekleştireceksiniz.

***Amaç dökümantasyonu:*** Proje ile ortadan kaldırmayı, iyileştirmeyi önerdiğimiz sorunlara işaret etmelidir. Bunun için Sorun analizi ve Hedef analizi yapmak önerilir. Amaç, Sorun ve Hedef analizini içermelidir.

Sorun Analizi: Projenin çözmekte rol oynayacağı/katkı sağlayacağı temel sorunun belirlenmesi ve alt sorunların da tanımlanmasıdır. “İlgili projenin temel sorunu ne?” sorusuna ve alt sorulara cevap arar.

Hedef Analizi: Sorun analizinde belirlenen temel sorunları ve alt sorunları olumluya çeviren önermeler ortaya koymaktır. Sorun çözümüne yönelik hedefler konulur.

**Projenin genel amacı**; Temel sorunun çözümüne yönelik bir hedef, **özel amaçları ise** alt sorunların çözümüne yönelik hedeflerdir. Özel amaçların proje sonunda gerçekleştiğinin ispat edilmesi gerekir.

# Özel amaçlar, Faaliyetler ile karıştırılmamalıdır. Örneğin bir eğitimin gerçekleştirilmesi, bir makinanın alınması FAALİYETTİR. Eğitim sonucunda katılımcıların teknik kapasite, bilgi ve becerilerinin arttırılması, alınan makine ile üretimde artış sağlamak ise faaliyete bağlı özel amaçlardır.

Araştırma Süreci:

Bir projede araştırma yapılmak istenirse süreçler sırasıyla;

* + Öncelikle literatür taraması yapılmalı. (Arama motorlarından şu ana kadar yapılan projeyle alakalı tarama yapmak)
  + Bu alandaki çalışmalar okunup notlar çıkarılmalı
  + Literatür taraması sonrası, araştırma sorusu belirlenmeli.
  + Veriler toplanmalı
  + Analiz yapılmalı
  + Analiz sonrası elde edilen bulgular yorumlanmalı
  + Son olarak proje raporu hazırlanmalı

Proje seçiminde cevap aranan sorular…

* + Hangi alanlara katkı sağlayacak
  + Ne düzeyde katkı sağlayacak
  + Ne kadar değerlidir
  + Ne kadar anlamlıdır
  + Hangi soruna/sorunlara nasıl çözümler bulmaktadır
  + Özgün ve yenilikçi yaklaşım var mıdır?
  + Kim/kimler için önemlidir. HEDEF KİTLESİ KİM?
  + Geliştirilebilir yanları var mıdır?

BAZI UML ve DİYAGRAM ÇİZİM ARAÇLARI

**Argo UML-Java tabanlı çizim aracı.** Tüm UML diyagramları (Sınıf diyagramları, Nesne diyagramları, Aktivity ve Sequence diyagramları) çizilebilir. Çizilen UML diyagramları için kod üretimi mümkündür. **Altova U Model:** Veri yönetimi, yazılım ve uygulama geliştirme, mobil ve veri entegrasyonu konularında yardımcı olan bir araç. Altova şirketi tarafından geliştirilmiş.

# Visual Paradigm

**IBM Rational Software Architect**

**Microsoft Office Visio.** Microsoft tarafından geliştirilmiş olan

**Online olarak** çizim yapılabilecek program (Bulut ortamında): **Gen My Model**

BAZI PROJE YÖNETİM ARAÇLARI

**Redmine:** Web tabanlıdır. Ruby on Rails web çatısı ile geliştirilmiş. Bünyesinde GANT (iş takip) çizelgesi barındırır.

# Taiga:

**Jira: Hata izleme ve Çevik (Agile) proje yönetimine izin veren, çevik metodolojiye uygun.**

**Apache Subversion:** 2000 yılında Collabnet firması tarafından başlanıp desteklenen bir sürüm kontrol sistemidir.

2

# Kullanılacak Yazılımlar

Proje raporlama ve dökümantasyon işlemlerinde ihtiyaç duyulabilecek bazı araçlar şöyledir; MS Office Word (proje raporu yazımı)

MS Office Excel (Tablolama vb.)

Adobe Photoshop (şekil çizimlerinde vb.)

IBM RSA(Retional Software Architect)(UML diyagram çizimlerinde) MS Visio(Akış şeması çizimlerinde vb.)

# Yazılım Geliştirme İstatistikleri

Tipik yazılım projesinin geliştirilmesi 1-2 yıl sürüyor ve en azından 500.000 kod satır içeriyor

Tüm projelerin yalnız %70-80’i başarıyla tamamlanıyor

Tüm geliştirme sürecinde her birey günde ortalama 10 satırdan az kod yazıyor

Geliştirme süresince her 1000 kaynak kod satırında 50-60 hata bulunuyor (satışa sunulmuş sistemde hata sayısı 4/1000’e düşüyor)

# Yazılım Problemleri(Krizleri)

Tasarlanan zamanın gerisinde kalma

Bütçeyi aşma

Düşük Kalite

Güvenilir olmayan yazılım



Kullanıcı taleplerinin karşılanmasında yetersizlik Sürekliliğin sağlanmasındaki zorluk

# Yazılım-Donanım Evrimi

Erken Yıllar 1950-1960 ilk bilgisayarlar, makine dili, 3-4 şirket

Yazılımlar; kullanıcı ile birebir iletişimde bulunmayan, işlerin toplu olarak verilip, yalnızca yazıcı çıktılarının alındığı biçimde geliştirilmekte idi.

Ayrıca, yazılımlar bu günkü gibi ürün tarzında değil, kuruluşa özel olarak geliştirilmekte idi.

# Programlama Dillerinin Seviyeleri 1.Kuşak- Makine Dili

10101110 10010001

**2. Kuşak- Assembler** 8085, Z80, 68000, vs. **3.Kuşak - Üst Seviye Diller** Pascal Coral66 Basic **Bildirimsel**

LISP Hope Prolog **Nesneye Yönelik Diller** Smalltalk, C++, Java

# Kuşak- Veri Yapısal

CICS, SQL

# Kuşak- Yapay Zeka Ve Paralel Programlama

CSP, OCCAM

**!** Tarihten bugüne süreç içerisinde donanım maliyetleri azalsa da, bakım ve idame maliyetleri artmaktadır. Geliştirme maliyetleri de zaman zaman geçmişle aynı maliyette veya artmaktadır.

Yazılım =Mantık(algoritma) +

Veri(test verisi, bilgi) + Belge (dokümanlar) +

İnsan (kullanıcı, geliştirici) + Program (kod)

“Bilgisayar sisteminin donanım bileşenleri dışında kalan her şey”

Yazılım

Mantık, veri, belge, insan ve program bileşenlerinin belirli bir üretim amacına yönelik olarak bir araya getirilmesi, ve yönetilebilmesi için kullanılabilecek ve üretilen, yöntem, araç, bilgi ve belgelerin tümünü içerir.

Mantık (algoritma)

Bilgisayarlaştırılmak istenen işin mevcut mantığı yazılıma yansıtılmak durumundadır.

Bu nedenle mantık (algoritma)bileşeni yazılımın en önemli bileşenlerinden biridir.

Veri

Her tür yazılım mutlaka bir veri üzerinde çalışmak durumundadır.

Veri dış ortamdan alınabileceği gibi, yazılım içerisinde de üretilebilir.

Yazılımın temel amacı veriyi bilgiye dönüştürmektir.

Belge (dokümanlar)

Yazılım üretimi bir mühendislik disiplini gerektirir.

Mühendislik çalışmalarında izlenen yol ya da kullanılan yaklaşımlar yazılım üretimi için de geçerlidir. Yazılım üretimi sırasında, birçok aşamada yapılan ara üretimlere ait bilgiler (planlama, analiz,

tasarım, gerçekleştirim, vb. bilgileri) belli bir düzende belgelenmelidirler.

**!** İşletmeler doğaları gereği dinamik bir yapıya sahiptir ve zaman içerisinde sürekli olarak yeni istek ve gereksinimler ortaya çıkabilmektedir. Bu yüzden yazılım geliştirirken “yeniden kullanılabilirlik” oldukça önemli bir unsurdur.

# Yazılım vs Donanım

Yazılım geliştirilir donanım üretilir.

Yazılım eskimez.

Yazılım en az donanım kadar önemlidir.

Yazılım kopyalama ve donanım kopyalama farklıdır.

***IEEE’nin yazılım mühendisliği tanımı***

*“Yazılım Mühendisliği:*

*Sistemli, düzenli, ölçülebilir bir yaklaşımın yazılım geliştirmede, yazılımın işlenilmesinde ve bakımında uygulanmasıdır.*

* D*iğer bir deyişle mühendisliğin yazılıma uygulanmasıdır.*

# Yazılım Hataları

|  |  |
| --- | --- |
| Mantıksal Tasarım | %20 |
| İşlevsel Tasarım | %15 |
| Kodlama | %30 |
| Belgeleme ve Diğerleri | %35 |

! Kalite sistemli bir proje, sanılanın aksine kalite sistemsiz bir projeye göre daha az masraflıdır.

# Yazılımda Hata Düzeltme Maliyetleri

Yazılım üretimindeki hatalar yayılma özelliği gösterir.

|  |  |
| --- | --- |
| Analiz | 1 |
| Tasarım | 5 |
| Kodlama | 10 |
| Test | 25 |
| Kabul Testi | 50 |
| İşletim | 100 |

# Yazılımda Kalite

Yazılımda kalite sağlama etkinlikleriyle

* Yazılım maliyetleri düşürülür
* Yazılım üretiminin yönetimi kolaylaşır
* Belgeleme ve standart sorunları giderilir.



3

* Yazılım Yaşam Döngüsü
  + Herhangi bir yazılımın, **üretim ve kullanım aşaması** birlikte olmak üzere geçirdiği tüm aşamalar biçiminde tanımlanır.
  + Yazılım işlevleri ile ilgili gereksinimler sürekli olarak değiştiği ve genişlediği için, söz konusu aşamalar bir döngü biçiminde ele alınır.
  + Müşteri istekleri sınırsız olduğu için, özellikle **BAKIM** aşaması bu döngüyü kendi içerisinde barındırır.
  + Döngü içerisinde herhangi bir aşamada geriye dönmek ve tekrar ilerlemek söz konusudur. Bu nedenle yazılım yaşam döngüsü **tek yönlü ve doğrusal değildir**. Tabi burada hangi geliştirme modeline göre işlettiğimiz de önemlidir.
  + **Yazılım yaşam döngüsü için oluşturulmuş standart**  IEEE 12207
* Yazılım Yaşam Döngüsü Çekirdek Aşamaları
  + **Planlama**  **Personel ve Donanım Gereksinimlerinin çıkartıldığı ve**

fizibilite(yapılabilirlik) çalışmasının yapıldığı aşamadır. PROJE PLANI

* + **Çözümleme(Analiz)**  **Sistem gereksinimlerinin çıkartıldığı aşama.2 Aşama vardır. 1- Mevcut sistemin analizi 2-Önerilen sistemin modellenmesi.**
  + **Tasarım**  **Yazılım sisteminin temel yapısının oluşturulduğu aşamadır.**
    1. **Mantıksal tasarım; önerdiğimiz sistemin yapısı anlatılır. Projenin gerçekleştirilmesine başlamadan önce mimari yapısının ortaya konulduğu faz.**
    2. **Fiziksel tasarım; Yazılımı içeren bileşenler ve bunların ayrıntılarını içerir.**
  + **Gerçekleştirim(implementasyon)**  **Kodlama, Test etme ve Kurulum çalışmaları.**
  + **Bakım**  (**Tüm döngüyü içerisinde barındırır**)  **Hata giderme ve yeni eklentiler yapma (teslimden sonra)**
* Yazılım Yaşam Döngüsü Ne Sağlar?
  + Mühendislik faaliyetlerinin yazılıma uyarlanması**(üretim, işletme ve bakım)**
  + Üretimden kullanıma kadar tüm sürecin planlanması ve tüm sürecin fazlara ayrılması  **(Yönetim Kolaylığı)**
  + Her bir aşamada ne yapılacak?
  + Her bir aşamada ki standartlara uyumluluk  **(Kaliteli yazılım)**
* Yazılım Yaşam Döngüsünün Gerçekleştirilmesi Amacıyla
  + **Belirtim (specification) yöntemleri**  Bir çekirdek sürece ilişkin fonksiyonları yerine getirmek amacıyla kullanılan yöntemler.
  + **Süreç (process) modelleri**  Yazılım yaşam döngüsünde belirtilen süreçlerin geliştirme aşamasında, hangi düzen ya da sırada nasıl uygulanacağını

tanımlayan modeller kullanılır.

* Belirtim Yöntemleri

1. **Süreç Akışı İçin Kullanılan Belirtim Yöntemleri**
   * Süreçler arası ilişkilerin ve iletişimin gösterildiği yöntemler 

(Veri akış Şemaları, Yapısal Şemalar, Nesne/Sınıf Şemaları)

1. **Süreç Tanımlama Yöntemleri**
   * Süreçlerin iç işleyişini göstermek için kullanılan yöntemlerdir.  (Düz Metin, Algoritma, Karar Tabloları, Karar Ağaçları, Anlatım Dili).
2. **Veri Tanımlama Yöntemleri**
   * Süreçler tarafından kullanılan verilerin tanımlanması için kullanılan yöntemlerdir.  (Nesne-İlişki Modeli, Veri Tabanı Tabloları, Veri Sözlüğü)

* **Yazılım Süreç Modelleri**
* **Yazılım üretim işinin genel yapılma düzenine ilişkin rehberlerdir**
* **Süreçlere ilişkin ayrıntılarla ya da süreçler arası ilişkilerle ilgilenmezler.**
  + **Gelişigüzel Model -****Öğrenci Ders Projeleri**
  + **Barok Modeli** **Kullanım kılavuzu gerektiren basit projeler**
  + **Çağlayan(Şelale) Modeli** **Hastane Sistemleri, Kontrol Sistemleri**
  + **V Modeli** **CRM (MÜŞTERİ İLİŞKİLERİ YÖNETİMİ) PROJE**
  + **Helezonik(Spiral) Model** **Askeri savunma yazılım sistemleri**
  + **Prototipleme** **Yüksek maliyetli projeler için prototipleme.**
  + **Evrimsel Model****Bankacılık uygulamaları**
  + **Artırımsal Model** **Öğrenci bitirme projeleri**
  + **Araştırma Tabanlı Model**  **Satranç, en büyük asal sayı**
* **Gelişigüzel Model**
  + **Herhangi bir model ya da yöntem yok.**
  + **Geliştiren kişiye bağımlı (belli bir zaman sonra o bile anlayamaz.)**
  + **Geliştirme güçlüğü yaşanabilir.**
  + **İzlenebilirliği ve bakımı oldukça zor.**
  + **1960’lı yıllarda kullanılan ilkel model**
  + **Genellikle tek kişilik üretim ortamı**
  + **Basit programlama**
  + **Yazılım yaşam döngüsünde ki fazlar da henüz belirgin değil.**
* **Barok Modeli**
  + **Yazılım yaşam döngüsü temel adımlarının doğrusal bir şekilde geliştirildiği model.**
  + **1970’li yıllarda kullanıldı.**
  + **Belgelemeyi ayrı bir süreç olarak ele alır, yazılımın geliştirilmesi ve testinden hemen sonra ele alır.**
  + **Halbuki, günümüzde belgeleme yapılan işin doğal bir ürünü olarak görülmektedir.**
  + **Aşamalar arası geri dönüşlerin nasıl yapılacağı tanımlı değil.**
  + **Gerçekleştirim aşamasına daha fazla ağırlık veren bir model olup, günümüzde kullanımı önerilmemektedir.**
* **Çağlayan (Şelale) Modeli**
  + **Yazılım yaşam döngüsü temel adımları baştan sona en az bir kez izlenerek gerçekleştirilir.**
  + **İyi tanımlı projeler ve üretimi az zaman gerektiren yazılım projeleri için uygun bir modeldir.**
  + **Geleneksel Model olarak da bilinen bu modelin günümüzde kullanımı giderek azalmaktadır.**
  + **Barok Modelin aksine belgeleme işlevini ayrı bir aşama olarak ele almaz ve üretimin doğal bir parçası olarak görür.**
  + **Barok modeline göre geri dönüşler iyi tanımlanmıştır.**
    - **Bu Modelin Avantajları(Çağlayan- Şelale-Geleneksel)**
      * **Müşteriler ve son kullanıcılar tarafından iyi anlaşılabilen adımlardan oluşur.**
      * **İterasyonlar (Tekrarlamalar) bir sonraki ve bir önceki**

adımlarla gerçekleşir, daha uzak adımların olması nadirdir.

* + - * **Değişiklik süreci yönetilebilir birimlere bölünmüştür.**
      * **Gereksinimleri iyi anlaşılabilen projelerde iyi çalışır.**
      * **Kalite gereksinimlerinin, bütçe ve zamana kıyasla çok daha önemli olduğu projelerde iyi çalışır.**
      * **Proje yöneticileri için kolaylıklar(işin dağılımını yapma vb.)**
    - **Bu Modelin Sorunları(Çağlayan-Şelale-Geleneksel)**
      * **Gerçek yaşamdaki projeler genelde yineleme gerektirir.**
      * **Genelde yazılımın kullanıcıya ulaşma zamanı uzundur.**
      * **Yanlışların düzeltilme ve eksiklerin giderilme maliyetleri yüksektir.**
      * **Projeyi yapan ekip mutsuz olur. Kod yazma dışında kalan(işin %80lik) kısmına ağırlık vermemektedirler.**
      * **Üst düzey yönetimlerin projeyi bir gider merkezi olarak görmeleri(Ürünü görme süresinin uzun oluşu nedeniyle)**
      * **Projenin bitmeyeceği düşüncesini yaygınlaştırır.**
* **V Süreç Modeli**
  + **Sol taraf üretim, sağ taraf sınama işlemleridir.**
  + **Her aşamada testi ön plana çıkaran bir model.**
    - **Kullanıcı Modeli**
      * **Geliştirme sürecinin kullanıcıyla olan ilişkileri**

tanımlanmakta. Sistemin nasıl kabul edileceğine ilişkin sınama belirtimleri ve planları ortaya çıkarılmaktadır.

* + - **Mimari Model**
      * **Sistem tasarımı ve oluşacak alt sistem ile tüm sistemin sınama işlemlerine ilişkin işlevler.**
    - **Gerçekleştirim Model**
      * **Yazılım modüllerinin kodlanması ve sınanmasına ilişkin fonksiyonlar gerçekleştirilir.**
  + **Belirsizliklerin az, iş tanımlarının belirgin olduğu BT projeleri için uygundur.**
  + **Bu model, kullanıcın projeye olan katkısını arttırmaktadır.**
  + **BT projesinin iki aşamalı olarak ihale edilmesi için oldukça uygundur;**
    - **İlk ihalede kullanıcı modeli hedeflenerek, iş analizi ve kabul sınamalarının tanımları yapılmaktadır.**
    - **İkinci ihalede ise ilkinde elde edilmiş olan kullanıcı modelleri tasarlanıp, gerçeklenmektedir.**
  + **V Süreç Modeli Avantajları;**
    - **Verification ve validation planları erken aşamada vurgulanır.**
    - **Verification ve validation sadece son üründe değil, tüm teslim edilebilir ürünlerde uygulanır.**
    - **Proje yönetimi tarafında takibi kolaydır.**
    - **Kullanımı kolaydır.**
  + **V Süreç Modeli Dezavantajları;**
    - **Aynı zamanda gerçekleştirilebilecek olaylara kolay imkan sağlamaz.**
    - **Aşamalar arasında tekrarlamaları kullanmaz. Risk çözümü yok.**
* **Helezonik (Spiral) Model**
  + **En önemli özelliklerinden biri RİSK ANALİZİ OLGUSUNU ön plana çıkarmak ve PROTOTİP yaklaşımını öngörmesi.**
  + **Her döngü bir fazı ifade eder. Doğrudan tanımlama, tasarım vs. gibi fazı yoktur.**
  + **Yinelemeli arttırımsal bir yaklaşım vardır.**

1. **Planlama: Üretilecek ara ürün için planlama, amaç belirleme, bir önceki adımda üretilen ara ürünle bütünleştirme.**
2. **Risk Analizi: Risk seçeneklerinin araştırılması ve risklerin belirlenmesi.**
3. **Üretim: Ara ürünün üretilmesi.**
4. **Kullanıcı değerlendirmesi: Ara ürün ile ilgili olarak kullanıcı tarafından yapılan sınama ve değerlendirmeler.**

* **Helezonik Model Avantajları;**
  + **Kullanıcı Katkısı**
    - **Yazılımı kullanacak personelin sürece erken katılması, ilerde oluşabilecek istenmeyen durumları engeller.**
  + **Yönetici Bakışı**
    - **Gerek proje sahibi, gerekse yüklenici tarafındaki yöneticiler, çalışan yazılımlarla süreç boyunca karşılaştıkları için, daha kolay izleme ve hak ediş planlaması yapılır.**
  + **Yazılım Geliştirici(Mühendis) Bakışı**
    - **Yazılımın kodlanması ve sınanması daha erken başlar.**
* **Prototipleme**
  + **Gereksinim tanımlama fazında hızlıca yapılan kısmi gerçekleştirme.**
  + **Gereksinimler netleştikçe prototip kullanıcı ihtiyacına göre düzeltilir.**
  + **Müşteri memnun olana kadar düzeltmelere devam et.**
* **Prototipleme – Avantajları**
  + **Kullanıcı sistem gereksinimlerini görebilir.**
  + **Karmaşa ve yanlış anlaşılmaları engeller.**
  + **Yeni ve beklenmeyen gereksinimler netleştirilebilir.**
  + **Risk kontrolü sağlanır.**
* **Prototipleme – Dezavantajları**
  + **Belgelenmesi olmayan hızlı ve kirli prototipler(quick and dirty)**
  + **Prototip hedefleri net değilse kod hackleme ya da jenga başlar.**
  + **Düzeltme aşaması atlanırsa düşük performansa yol açar.**
  + **Müşteri prototipten de son ürün gibi görünüm ve etki bekler.**
* **Evrimsel Geliştirme Süreç Modeli**
  + **İlk tam ölçekli modeldir.**
  + **Coğrafik olarak geniş alana yayılmış, çok birimli organizasyonlar için önerilmektedir. (Banka uygulamaları)**
  + **Her aşamada üretilen ürünler, üretildikleri alan için tam işlevselliği sahiptir.**
  + **Pilot uygulamada kullan, test et, güncelle diğer birimlere taşı**
  + **Modelin başarısı ilk evrimin başarısına bağımlıdır.**
  + **İki çeşit evrimsel geliştirme vardır.**
  1. **Keşifçi Geliştirme (Exploratory Development)**
     + **Hedef: Müşterinin gereksinimlerini incelemek için müşteri ile çalışıp son sistemi teslim etmek**
     + **İyi anlaşılan gereksinimlerle başlanmalıdır.**
     + **“Ne istediğimi sana söyleyemem ama onu gördüğümde bilirim.” Yaklaşımı.**
  2. **Atılacak Prototipleme (throw-away prototyping)**
     + **Hedef: Sistem gereksinimlerini anlamak.**
     + **Tam anlaşılmamış gereksinimlerle başlar.**
* **Aksaklıkları(Evrimsel Model)**
  + **Değişiklik denetimi**
  + **Konfigürasyon yönetimidir.**
  + **Kalite yönetimi**
  + **Sürüm yönetimi**
* **Artırımsal Geliştirme Süreç Modeli**
  + **Üretilen her yazılım sürümü birbirini kapsayacak ve giderek artan sayıda işlev içerecek şekilde geliştirilir.**
  + **Öğrencilerin bir dönem boyunca geliştirmeleri gereken programlama ödevlerinin 2 haftada bir gelişiminin izlenmesi( Bitirme tezleri)**
  + **Uzun zaman alabilecek ve sistemin eksik işlevlikle çalışabileceği türdeki projeler bu modele uygun olabilir.**
  + **Bir taraftan kullanım, diğer taraftan üretim yapılır.**
* **Araştırma Tabanlı Süreç Modeli**
  + **Yap – At prototipi olarak da bilinir.**
  + **Araştırma ortamları bütünüyle belirsizlik üzerine çalışan ortamlardır.**
  + **Yapılan işlerden edinilecek sonuçlar belirgin değildir.**
  + **Geliştirilen yazılımlar genellikle sınırlı sayıda kullanılır ve kullanım bittikten sonra işe yaramaz hale gelir ve atılır.**
  + **Model – Zaman – Fiyat kestirimi olmadığı için sabit fiyat sözleşmelerinde uygun değildir.**
* **Örnek Olarak**
  + **En hızlı çalışan asal sayı test programı**
  + **En büyük asal sayıyı bulma programı**
  + **Satranç programı**
* **Metodolojiler**
* **Metodoloji: Bir BT Projesi ya da yazılım yaşam döngüsü aşamaları boyunca kullanılacak ve birbirleriyle uyumlu yöntemler bütünüdür.**
* **Bir metodoloji;**
  + **Bir süreç modelini,**
  + **Belirli sayıda belirtim yöntemini içerir.**
* **Günümüzdeki metodolojiler genelde Çağlayan ya da Helezonik modeli temel almaktadır.**
* **Bir Metodojide Bulunması Gereken Temel Bileşenler**
* **Metodoloji bileşenleri ile ilgili olarak bağımsız kuruluş (ISO, IEEE vs.) ve kişiler tarafından geliştirilmiş çeşitli standartlar ve rehberler mevcuttur.**
* **Standart ve rehberler sürekli olarak güncellenmektedir.**
* **Bir kuruluşun kendi metodolojisini geliştirmesi oldukça kapsamlı, zaman alıcı ve uzmanlık gerektiren bir faaliyet olup. İstatistikler 50KİŞİ/AY ‘LIK bir iş gücü gerektiğini göstermektedir.**
  + **BİR METODOLOJİ ÖRNEĞİ**
    - **Yourdan yapısal sistem tasarımı metodolojisi;**
    - **Kolay uygulanabilir bir model. Günümüzde yaygın.**
    - **Çağlayan modelini temel almaktadır.**
    - **Bir çok CASE aracı tarafından doğrudan desteklenmektedir.**

\*\*\*4. HAFTA\*\*\*

**Günümüzde yazılım projelerinin durumu:** %4-5 oranında tam başarı

# Başarısızlığın ana sebepleri:

* Müşterinin isteklerini doğru analiz edememek

|  |  |  |
| --- | --- | --- |
| * Proje için uygun ekibi kuramamak |  | |
| * Yanlış teknoloji ve mimari seçimleri |
| * Geleneksel yöntemlerin eksiklikleri |
| * Müşteriyle iletişimden kaçınmak vs. |
| %26: Vizyon eksikliği  %24: Değişime direnç  %17: Planlama hataları  %10: Yanlış teknolojiler  %9: Ekip değişiklikleri  %8: Yanlış hedefler  %3: Bütçe hataları  %3: Diğer |
| \*Tüm bunlar çevik yazılımla çözülebilir |
| **Çevik Yazılım Yöntemi: Çok önemsediği noktalar:** | **/ / /** | **Az önemsediği noktalar:** |
| Bireylerle Etkileşim Çalışan Bir Yazılım Müşterilerle İşbirliği  Değişikliklere Uyum Sağlama |  | Süreç Ve Araçlar  Detaylı Belgelendirme Sözleşmedeki Kurallar Belirli Bir Plan |

* Hızlı, devamlı ve kullanışlı yazılım üreterek müşteri memnuniyeti sağlamayı amaçlar.
* Geliştiriciler ile iş adamları arasında günlük ve yakın işbirliği bulunmalıdır.
* Çalışan yazılım gelişimin en önemli ölçüsüdür.
* Taleplerdeki geç değişikliklerin de memnuniyetle karşılanır.
* Yüz yüze görüşme iletişimin en güzel yoludur.
* Kendi kendini organize eden takım yapısı gereklidir.
* Basitlik önemlidir.

# Temel Prensipler:

* Müşteriyi memnun etmek
* Değişen ihtiyaçları karşılamak
* Sık aralıklarla ürün teslimi yapmak
* Yüz yüze iletişime önem vermek
* Sürdürülebilir gelişmeyi desteklemek
* Teknik mükemmeliyete, iyi dizayna ve sadeliğe odaklanmak
* Kendi kendine organize olan takımlar kurmak

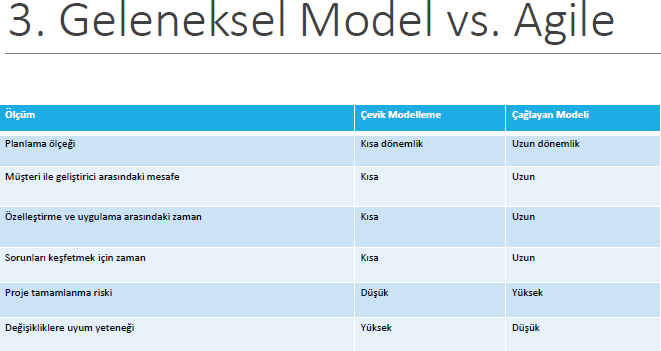
Çevik model takımları; kendi kendilerine organize olmuş, hedefleri net, işlerine odaklı, teslim edilebilecek düzeyde ürün ortaya koyabilen 3-7 kişilik gruplardan oluşur. Özellik bazlı takım oluşturulur.

# Geleneksel model vs. Agile

**Geleneksel model:** Müşteriler ne istediğini iyi bilir. Geliştiriciler neyi ne şekilde üreteceklerini iyi bilir. Bu yol boyunca hiç bir şey değişmeyecektir.

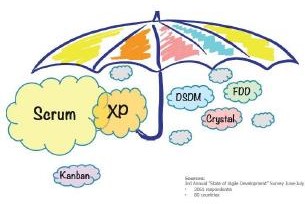
**Agile (Çevik):** Müşteriler ne istediğini keşfeder.Geliştiriciler neyi nasıl üreteceğini keşfeder. Bu yol boyunca birçok değişiklik yapılabilir.

\*Geleneksel yöntemde başarı oranı %14 iken Çevik yöntemde %42’dir.



# 3 somut değişiklik yapın>

1. Gerçek takımlar oluşturun
2. Sık sık teslimat yapın (projenin tüm çeyrekleri sonunda)
3. Gerçek kullanıcıları dahil edin.



**FDD:** Feature – Driven Development

**RUP:** Rational Unified Process

**DSDM:** Dynamic System Devolopment Method

**FDD (Özellik güdümlü geliştirme):** Basit iyi tanımlanmış sistem iyi çalışır. Süreç adımları basit olmalıdır. Sistemi hazırlamak için gereken sistem büyüyebilir olmalıdır. Büyük projeler için de kullanılabilir olmalıdır. İyi süreç arka plana saklanır ve insanlar sonuçlara odaklanabilir. Kısa, iteratif, özellik yaklaşımlı yaşam döngüleri en iyi sonucu verir.

**RUP:** IBM tarafından oluşturulan bir iteratif yazılım geliştirme süreçlerinden biri. Tümleşik modelleme yapısına uygun. Başarısız bir yazılımdaki sorunların aşılıp başarılı yazılım oluşturmak için gerekli adımları saptayarak oluşturulmuş bir süreçtir.

* Müşteriyi ve yazılımcıyı organize edebilmesi,
* Standart tanımlı adımları olması,
* Oluşacak yazılımdaki sık değişiklikleri öngörebilmesi,
* Basit olması,
* Proje yönetim aktivitelerinin çok fazla olmaması. (Sadece geliştirme ön plana çıkar.)

**XP (Extreme Programming -- Uç Programlama):** yazılım geliştirme süreci boyunca son derece kaliteli olmak koşuluyla çalıştırılabilir kod üretmeye odaklanmış bir yazılım geliştirme metodolojisidir. Sürecin en başından itibaren çalıştırılabilir kodu sürecin merkezinde

tutmaktadır.

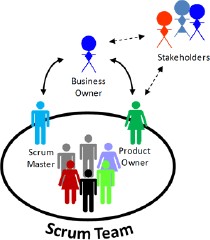
* Planlama
* Sık ve küçük sürümler
* Basit tasarım:
* Önce test
* Refactoretme
* Haftada 40 saat çalışma
* Müşteriyle yakın iletişim
* Kodlama standartları

# ~SCRUM MODELİ~

Scrum, çok karmaşık ürün yönetimini basite indirgemek için yapılandırılmış bir yöntemdir. Takım yönetiminde geliştiriciye görünür olması gerek. Şeffaf yönetim söz konusu. Takım yönetiminde ortak bir görüş ortaya koyulur. Kullanıcılarına yönelik istenmeyen sapmaları

tespit etmek için sürecin ve programın sürekli denetim altında tutulması gerekiyor.

Scrum takım modeli esneklik, yaratıcılık ve verimliliği optimize etmek için tasarlanmıştır. Takım kendi kendini örgütler.



**Scrum Master:** Takım lideri. Uzun vadeli ürün planlamaları. Takımın verimliliğini arttırmakla görevlidir. Gün başında 15dk’ lık toplantılar düzenler.

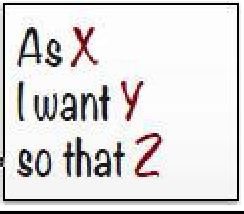
**Product Owner:** Ürünün geri bildirimlerinden (backlog) sorumlu. Geliştirme ekibi onun kararlarına göre hareket eder.

# Backlog:

* Müşteriden ve son kullanıcıdan gelen gereksinimleri içerir.
* "Ne yapacağız" sorusunun yanıtını içerir.
* Herkese açık ve herkes tarafından müdahale edilebilir.
* Risk, iş değeri, zaman gibi kavramlara göre ürün sahibi tarafından sıralandırılır.
* User Story' lerden oluşur.

**Sprint:** Bir aylık proje döngülerine verilen isimdir. Sprint süresi boyunca her gün toplantılar yapılır. Toplantılarla içerik belirlenir. Sonunda ortada değeri olan bir çıktı olmalıdır.

**User Story:** Müşteri, son kullanıcı veya ürün sahibi için değerli olan ve anlam ifade eden genellikle fonksiyonel özelliklerin belirtildiği ifadelerdir.



**X** kişisi olarak

**Z** olsun diye

**Y** ‘yi istiyorum

**Örnek User Story:** Online alışveriş yapan biri olarak, alışverişe daha sonra devam edebileyim diye, alışveriş kartımın kaydedilmesini istiyorum.

|  |  |
| --- | --- |
| **SCRUM** | **XP (EXTREMEPROGRAMMİNG)** |
| Sprint (2 hafta – 1 ay) | Sprint (1 ya da 2 hafta) |
| Sprintler en son halini aldıktan, toplantı yapıldıktan sonra değişmez. | Sprintler değişebilir. |
| Özellikler geliştiriciler tarafından derecelendirilir. | Özellikler ürün sahibi tarafından derecelendirilir. |
| Herhangi bir mühendislik pratiği tanımlamaz. | Mühendislik pratikleri tanımlar. Eşli programlama, otomatik test, basit dizayn vs. |

**5**

5.HAFTA

\*En Önemli aşamalardan birisidir(planlama)

\*Yazılım geliştirme sürecinin ilk aşaması

\* Proje planlama aşamasında yapılan işlemler 1.Proje Kaynaklarının belirlenmesi

1. Proje maliyetlerinin kestirilmesi
2. Proje ekip yapısının oluşturulması
3. Ayrıntılı proje planı yapılması
4. Projenin izlenmesi

\*Proje planı tüm proje süresince kullanılabilecek gözden geçirilebilecek ve güncellenecek bir belgedir 1.PROJE KAYNAKLARI

\*Zaman kullanımını görev sürelerini edinilme zamanlarını planlar ve 3 grupta incelenir

* 1. İnsan kaynakları : elemanların hangi süreyle ve hangi aşamada olacağını belirler
  2. Donanım kaynakları : ana bilgisayarlar, sunucular(hep, e-posta, veritabanı ) kurumsal firmada bu sunucular farklı makinelerden nedeni a trafiği engellemek için PC LAN WAN
  3. Yazılım kaynakları : bilgisayar destekli tasarım (CAD)ve bilgisayar destekli mühendislik (CASE) araçları olarak bilinmektedir

\*İş sistemleri planlama araçları

\*Proje yönetim araçları

\*Analiz ve tasarım araçları

\*Programlama araçları

\*Test araçları

\*Simülasyon araçları

\*Bakım araçları ve destek araçları

2 PROJE MALİYETLERİNİN KESTİRİLMESİ

\*Bir bilgi sistemi yada yazılım için gerekebilecek işgücü ve zaman maliyetlerinin üretiminden önce belirlenebilmesi için yapılan işlemlerdir

KULLANILAN UNSURLAR

\*Geçmiş projelere ilişkin bilgiler

\*Proje ekibinin deneyimleri

\*İzlenen geliştirme modeli

MALİYET YÖNETİMİ SAYESİNDE;

\*Gecikmeler önlenir

\*Bilgi sistemi geliştirme süreci kolaylaştırılır

\*Daha etkin kaynak kullanımı sağlanır

\*İş Zaman planı etkin olarak gerçekleştirilir

\*Ürün sağlıklı olarak fiyatlandırılır

\*Ürün zamanında ve hedeflenen bütçe sınırları içerisinde bitirilir GÖZLEMLENEBİLECEK DEĞERLER;

\*Projenin toplam süresi

\*Projenin toplam maliyeti

\*Projede çalışan eleman sayısı

\*Toplam satır sayısı

\*Bir satırın parasal maliyeti

\*Bir kişinin ayda gerçekleştirdiği satır sayısı

\*Toplam işlev noktası sayısı

\*Bir işlem noktasının parasal maliyeti

\*Bir ayda gerçekleştirilen işler noktası sayısı

\*Bir kişinin bir ayda gerçekleştirdiği işler noktası sayısı

\*Kişinin maliyeti

İŞLEV NOKTALARI YÖNTEMİ

İşlev noktaları, geliştirmenin erken aşamalarında saptanan bir değerdir ve 3 temel adımda izlenir

1. problemin bilgi ortamının incelenmesi
2. Problemin teknik karmaşıklığın incelenmesi
3. İşlev noktası hesaplama

1 .PROBLEMİN BİLGİ ORTAMININ İNCELENMESİ

Bu aşamada problemle ilgili temel girdiler, çıktılar, kütükler, sorgular ve arayüzler incelenir problemin bilgi ortamı, 5 ana bileşenden oluşur

\*Kullanıcı girdileri: yazılıma girdi olarak verilen her farklı uygulama bileseni bir kullanıcı girdisi olarak sayılır kullanıcı girdilerini sayısı hesaplanırken alan bazında değil daha genel olarak mantıksal kayıt bazında uygulama yapılmalıdır

Örneğin bir personel sisteminin geliştirilmesinden söz edildiğinde personel sicil bilgilerini personel izin bilgilerini gibi bilgiler aynı kullanıcı girdiler olarak sayılabilir

\*Kullanıcı Çıktıları : kullanıcıyı ilgilendiren her türlü mantıksal çıktı kullanıcısı olarak sayılmalıdır bu kapsamda kullanıcı çıktılarını örnek olarak raporlar ekran çıktıları hata iletileri ve benzeri verilebilir

girdiler gibi çıktılar da genel olarak sayılmalıdır bir raporun içerisindeki bir alan kullanıcı çıktısı olarak sayılmamalıdır öte yandan aynı bilgileri içeren ancak bilgi düzeni ya da sıralaması farklı olan

raporlarda ayrı kullanıcı çıktısı olarak ele alınmamalıdır

Örneğin soyadına göre sıralanmış personel listesi ve sicil nosuna göre sıralanmış personel listesi raporları farklı kullanıcı çıktılar olarak sayılmamalıdır

\*Kullanıcı sorguları: kullanıcı sorgusu çevrimiçi olarak bilgisayarlara verilen bir kedi sonuncu yine

çevrimiçi olarak bir kullanıcı çıktı saldırması biçiminde tanımlanmaktadır her farklı sorgu ayrı olarak sayılmalıdır

Örneğin personel sicil bilgilerinin sorgulanması personel maaş bilgilerinin sorgulanması vb.

\*Kütükler: her mantıksal bilgi yıgini ya da kütük ayrı olarak sayılmalıdır. Personel sicil kütüğü, personel maaş kütüğü personel hareket kütüğü vb.

\*Arayüzler: geliştirilmesi öngörülen bilgi sisteminin, gerek kurum içinde gerekse kurum dışında bir başka bilgi sistemi ile bilgisayar ortamında bir iletişimi söz konusu ise bu durum bir dışsal arayüz olarak sayılmalıdır kağıt ya da rapor türündeki arayüzler bir dışsal arayüz sayılmazlar

Örneğin, personel Bilgi sistemi, yine aynı kurumda olduğu varsa elden muhasebe bilgi sistemine, personel maaşlarına ilişkin bilgiyi bilgisayar ortamında aktarılması bir dışsal arayüz olarak sayılabilir

\*Problem bilgi ortamı için elde edilen değerler , karmaşıklığı düzeylerine göre seçilecek karmaşıklık değeri ile çarpılır ve çarpımlarının sonucu elde edilen toplam işlem nokta sayısı, ayarlanmış işlevnokta sayısı (AİN) olarak adlandırılır

2. PROBLEM TEKNİK KARMAŞIKLIĞININ İNCELENMESİ

Bu adımda, probleme ilişkin dışsal etkenlerin değerlendirilmesi yapılır problem teknik karmaşıklık faktörü problemin karmaşıklığına ilişkin sorulara verilecek yanıtların toplamı ile elde edilir uygulama ortamının ve uygulamanın özellikleri dikkate alınarak her soruya 0 ile 5 arasında bir yanıt verilir(TKF) 3.İŞLEV NOKTA SAYISI HESAPLAMA

İN =AİN\*(0.65\*0.01\*TKF)

İŞLEV NOKTA SAYISI ÇEŞİTLİ AMAÇLARLA KULLANILABİLİR ÜRETKENLİK = İN/KİSİ-AY

KALİTE=hatalar/İN

ETKİN MALİYET MODELİ COCOMO

Tüm COCOMO modelleri, temel girdi olarak satır sayısı kestirimini alır ve çıktı olarak işgücü ve zaman çıktılarını verir.

İş gücü K =a\*S^b Zaman=c\*K^d

a,b,c,d: her biri model için farklı olan katsayılar S=1000 türünden satır sayısı

Öte yandan COCOMO modelleri farklı türdeki yazılım projelerinin farklı biçimde uygulanır projeler 3 biçimde sınıflandırılmaktadır

1. AYRIK PROJELER

Deneyimli ve küçük proje ekipleri tarafından geliştirilecek, boyutları küçük bilgi sistemi

projeleri dir bu tür projelerde, Bir donanımın yazılım tarafından sürülmesi ya da yönetilmesi söz konusu değildir genellikle, yerel ağ üzerinde çalışan, girdi ve çıktıları tümüyle veri bilgi

tabanlı olan projelerdir örneğin, bir kurumun yerel ağ üzerinde çalışan insan kaynakları yönetim sistemleri vb.

K=2.4\*S^1.05 işgücü T=2.5\*K^0.38

1. YARI GÖMÜLÜ PROJELER

hem bilgi boyutu hem de donanım sürme boyutu olan projelerdir

1. GÖMÜLÜ PROJELER

Donanım sürmeyi hedefleyen ve donanım kısıtları yüksek projelerdir (pilotsuz uçağı yönlendiren yazılım projeleri vb.

PROJE PLANI

1. PROJE KAPSAMI: geliştirilecek sistemin genel tanımını ve sınırlarını içerir yalnızca neyin yapılacağı değil nelerin yapılmayacağı da proje kapsamında yer almalıdır
2. PROJE ZAMAN-İŞ PLANI : projede yapılacak kişilerin iş adımları, ara ürünlerin neler olacağı zamanlaması ve ne zaman üretileceği, kimler tarafından üretileceği türündeki bilgiler proje zaman iş planında yer alır bu plan proje geliştirme döngüsü sürecinde neler yapılacağını zaman boyutuyla içerir proje zaman iş planını hazırlamak amacıyla geliştirilmiş birçok araç ve yöntem bulunmaktadır bu araçlardan en yaygın olarak kullanilanına örnek olarak MS project verilebilir
3. PROJE EKİP YAPISI: proje oluştururken ne tür insan kaynaklarına ihtiyaç var bu kısımda bunlar belirlenir
4. ÖNERİLEN SİSTEMİN TEKNİK TANIMLARI: proje planında ,projede kullanılacak donanım, altyapısı, işletim sistemi, veri tabanı yönetim sistemi vb. Yazılımlara ilişkin özellikler önerilen sistemin teknik tanımları bölümünde yer alır varsa, özel amaçlı donanımlar ve bunların

çizimleri, kablolama, işletim zamanı kısıtları gibi gerekler ayrıca belirtilir

1. PROJE STANDARTLARI , YÖNTEME METODOLOJİLERİ: projenin geliştirilmesi amacıyla kullanılacak standartlar, yöntemlerle mitolojiler olacak yanında yer almalıdır söz konusu standart, yöntem ve metodolojileri örnek olarak

\*çözümleme aşamasında kullanılacak yöntemler

\*Tasarım aşamasında kullanılacak yöntemler

\*Belgeleme standartları ve tasarım standartları vb. Verilebilir bu konuda en önemli kaynaklardan biri İEEE standart ve rehberleredir

1. KALİTE SAĞLAMA PLANI: projede yapılan işlerin doğruluğunun denetlenmesi, proje standartları ve izleme yöntemlerinin oluşturulması, proje kalitesinin sağlanması amacıyla

kalite sağlama planı oluşturulur bu plan ayrıca projede oluşturulacak kalite yönetim biriminin görev ve iş tanımlarını yapar ana ilke olarak her ara ürün ancak kalite denetiminden geçtikten sonra iş sahibi ya da müşteriye verilir

1. KONFİGÜRASYON YÖNETİM PLANI: sistemin geliştirilmesi sırasında herhangi bir aşamada oluşabilecek değişikliklerin kaydedilmesi, izlenmesi için kullanılacak yöntem yönergeler

konfigurasyon yönetim planında yer alır bu planda temelde müşteri ya da bir iş sahibine değişikleri bildirme ve izleme konusunda yardımcı olması amacıyla hazırlanır

1. KAYNAK YÖNETİM PLANI: projenin , geliştirme süresi boyunca fiziksel kaynakların insan kaynaklarının ve finansal kaynakların yönetimi sürekliliğini sağlama biçimi ile ilgili konular kaynak Yönetim planı içerisinde yer alır
2. EĞİTİM PLANI: büyüklüğü ne olursa olsun her projede eğitim boyutu içerir bir proje için sağlanacak en düşük düzeydeki eğitim geliştirilen uygulama yazılımları ile ilgili olarak uç

kullanıcıların eğitimidir bu eğitimin yanısıra bir proje için gerekebilecek eğitimlere örnek olarak kullanıcılar için temel bilgisayar kavramları Windows Office yazılımları teknik eğitim eğitimleri verilebilir söz konusu eğitimlerin ne zaman kime hangi ortamda verileceği gibi bilgiler eğitim planında yer alır

1. SINAMA PLANI: her proje planı geliştirme süresinde üretilecek ürünün gelecek teknik açıdan gerekse kullanıcı açısından sınanacağına alınacağına ilişkin bilgileri icermek durumundadır

aksi durumda projenin sonlandırılması olası olmaz söz konusu bilgiler proje sınama planı içerisinde yer alır

1. BAKIM PLANI: proje uygulama aşamasını geçtikten sonra gerekebilecek değişiklikleri ne zaman ve nasıl yapılacağını içerir

6

SİSTEM ÇÖZÜMLEME

SİSTEM ÇÖZÜMLEMEYE GİRİŞ

-Üretim sürecinin başlangıcıdır. Mevcut sitemin nasıl çalıştığı araştırılır.

-Mevcut sistemin incelenmesi aşamasında temel hedef gereksinimlerin saptanmasıdır. Bu işlemden sonra sistem için modelleme yapılır. Bu modele mantıksal model adı verilir.

-Çözümleme aşamasında iki durum söz konusudur; 1)Mevcut sistemin incelenmesi

2)Önerilen sistemin modellenmesi

-Çözümleme çalışmasında mutlaka bir model takip edilmelidir. Aksi takdirde süreç dağınık bir biçimde sürer ve denetlenemez.

-Bu bölümde yapısal sistem geliştirme yaklaşımında kullanılan yöntemlerden en yaygın olanlarına örnekler verilmiştir.

-Yöntemler veri modelleme ve süreç modelleme olarak iki başlık altında incelenecektir. GEREKSİNİM NEDİR?

-Bir sistem geliştirirken kullanıcının sistemin işlevleri ile ilgili beklentileri sistemin amaçlarını oluşturur.

-Bu amaçların yerine getirilmesine de gereksinim adı verilir.

-Bir kuruluş için personel bordro sistemi geliştirdiğimizi varsayalım. Maaş formlarının aylık olarak hazırlanması, kuruluşun değişik birimlerinden bu bilgilere erişim istekleri gereksinimlere örnektir.

GEREKSİNİM TÜRLERİ 1)İŞLEVSEL GEREKSİNİMLER

-Sistem ile çevresi arasındaki iletişimi belirleyen gereksinimler olarak tanımlanır. Sistemin herhangi bir durum karşısında durumunu belirleyen fonksiyonel gereksinimler bu başlık altında incelenmektedir.

-Örneğin: Maaş bordro sisteminde; maaş çekinin ne zaman hazırlanacağı, çek hazırlamak için ne tür girdiler verilmesi gerektiği vb.

2)İŞLEVSEL OLMAYAN GEREKSİNİMLER

-Kullanıcının sorununun bağımsız olarak çözülmesi gereken sorunlar işlevsel olmayan gereksinim olarak tanımlanır.

-Daha çok sisteme dair kısıtları ifade eden işlevsel olmayan gereksinimler başlığı altında işlenen ihtiyaçlardır.

-Örneğin; kullanılacak bilgisayarın türü, kullanılacak yazılım geliştirme ortamı, veri tabanı yönetim sistemi vb.

GEREKSİNİMLER 1)FİZİKSEL ÇEVRE

-Fiziksel çevre gereksinimlerinin belirlenmesi için önemlidir.

-Örneğin; işlevlerin geliştirileceği aygıtlar nerededir?

-Tek bir yerde mi olacak birden çok fiziksel birbirinden ayrılmış yerler söz konusu mu?

-Sıcaklık, nem oranı veya manyetik etkileşim gibi çevresel kısıtlamalar var mı?

2)ARAYÜZLER

-Girdiler bir mi yoksa birden çok sistemden mi geliyor?

-Çıktılar bir mi yoksa birden çok sisteme mi gidiyor?

-Verinin nasıl biçimlendirileceğine ilişkin önerilen bir yol var mı?

-Verinin kullanılacağı, önerilen bir ortam var mı?

-Arayüzlerin kullanımı kolay m? 3)KULLANICI VE SİSTEM ETMESİ

-Sistemi kim kullanacak?

-Çeşitli kullanıcılar olacak mı?

-Her bir kullanıcı tipinin yetenek düzeyi nedir?

-Bir kullanıcının sistemi anlaması ne denli kolay olacaktır?

-Bir kullanıcının sistemi kötü amaçla kullanması ne kadar zordur? 4)İŞLEVSELLİK

-Sistem ne yapacak?

-Sistem yapacağı işi ne zaman gerçekleştirecek?

-Sistem nasıl ve ne zaman değiştirilebilir veya güçlendirilebilir?

-Çalışma hızı, yanıt süresi ya da çıktı üzerinde kısıtlayıcı işlemler var mı? 5)BELGELEME

-Ne kadar belgelemenin gerektiği?

-Belgeleme hangi kullanıcı kitlesini hedeflemektedir? 6)VERİ

-Hem giriş hem çıkış için verinin biçimi ne olmalıdır?

-Bu veri kesinlik ölçüsü ne olmalıdır?

-Bu veri ne sıklıkta alınacak veya gönderilecektir?

-Hesaplamalar hangi duyarlılık derecesine kadar yapılacaktır?

-Belli bir zaman süresince veri kaynağında saklanacak mıdır? 7)KAYNAKLAR

-Sistemi kurmak için, kullanılacak ve bakımını yapmak için ne kadar malzeme, personel ve diğer kaynaklara ihtiyaç vardır?

-Geliştirici hangi yeteneklere sahip olmalıdır?

-Sistem ne kadar fiziksel yer kaplayacak?

-Güç, ısıtma ve soğutma için gereksinimler nelerdir?

-Geliştirim için tavsiye edilen bir zaman çizelgesi var mıdır?

-Geliştirme veya yazılım ve donanım için harcanacak para üzerinde bir sınırlama var mıdır?

1. GÜVENLİK

-Sisteme veya bilgiye erişim denetlenmeli midir?

-Bir kullanıcının verisi diğerlerininkinden nasıl ayrıştırılacak?

-Kullanıcı programları diğer program ve işletim sistemlerinden nasıl ayrı tutulacak?

-Yedek kopyaları başka bir yerde mi saklanmaktadır?

-Yangın ve hırsızlığa karşı ne tür önlemler alındı? 9)KALİTE GÜVENCESİ

-Güvenlik için gereksinimler nelerdir?

-Sistemin özellikleri insanlara nasıl aktarılmalı?

-Sistem çökmeleri arasında öngörülecek zaman nedir?

-Bir çöküm sonrasında istemi tekrar başlatmak için uygun görülebilecek bir süre var mıdır?

-Sistem tasarımda yapılacak değişiklikleri nasıl özümseyecek?

-Bakımdan anlaşılan sadece hata gidermek midir yoksa sistemi geliştirmeyi de içerir mi?

-Kaynak kullanımı ve yanıt süresine ilişkin verimlilik ölçütlerinelerdir?

-Sistemi bir yerden bir yere veya bir bilgisayardan diğerine taşımak ne kadar kolaylıkta olmalıdır? GEREKSİNİM ÖZELLİKLERİ

* Gereksinimler sadece bir sisteme ve sisteme yapılan veri dönüşümünü tanımlaması, aynı zamanda sistemin performansı üzerindeki kısıtları da tanımlar.
* Bu yüzden gereksinimler üç aşamada hizmet eder.
* Birincisi; geliştiricilerin, müşterilerin sistemin nasıl çalışmasını istediklerini anlamalarını sağlar.
* Gereksinimler ayrıca tasarımcılara, sonuç sistemin ne işlevsellik özellikte olacağını söyler.
* Üçüncüsü; gereksinimler, sınama ekibine, kullanıcıyı sunulan sistemin istenen sistem olduğuna ikna etmek için neler göstermeleri gerektiğini söyler.

Hem geliştiriciler hem de kullanıcılar gereksinimleri kullandığı için, gereksinimlerin geçerliliğinin doğrulanması gerekir. Doğrulanma süreci aşağıda belirtilen yedi kriterleri içerir;

* 1. Gereksinimler doğru oluşturulmuş mudur?

-Hem geliştirici hem de kullanıcı tarafından belirsizliği giderilmiş hatasız olarak onaylanmış olduğunu güvenceye almak için gözden geçiriniz.

* 1. Gereksinimler tutarlı mı?

-Yani, hiç çelişen, çift anlamlı, anlaşılmayan gereksinimler var mı?

* 1. Gereksinimler tam mı?

-Eğer tüm olası durumlar bir gereksinimle açıklanıyorsa, gereksinimler tamdırlar. Bu nedenle örneğin bir çalışan ücretsiz izne çıktığında, birisi terfi ettiğinde veya bir kişi avans istediğinde ne olacağı örneğin bir personel sistemi tarafından açıklanmalıdır. Tanımlama, kullanıcı tarafından

istenen tüm özellikleri içeriyorsa bir sistem için dışsal tamdır denir. Bir tanımlama; eğer gereksinimler içerisinde açıklanmamış referanslar yoksa içsel tam olarak tanımlanır.

* 1. Gereksinimler gerçekçi mi?

-Kullanıcının istediği gerçekten yapılabilir mi? Geliştirme zamanının uzun olduğu durumlarda teknolojik gelişime uyabilecek biçimde tanımlı mı?

* 1. Her gereksinim kullanıcı tarafından istenen bir şeyi mi tanımlamaktadır?

-Bazı durumlarda bir gereksinim bizi gereksiz yere kısıtlayabilir. Örneğin kullanıcı ünü iyi olduğu için XYZ bilgisayarının kullanılmasını isteyebilir. Öte yandan XYZ istenen sistemin gerçekleştirilmesinde en uygun bilgisayar olmayabilir. Bu tür gereksinimler doğrudan sistemin hedeflerine adreslemez. Yalnızca işe yarayacak gereksinimler üzerinde yoğunlaşmalıdır.

* 1. Gereksinimler doğrulanabilir mi?

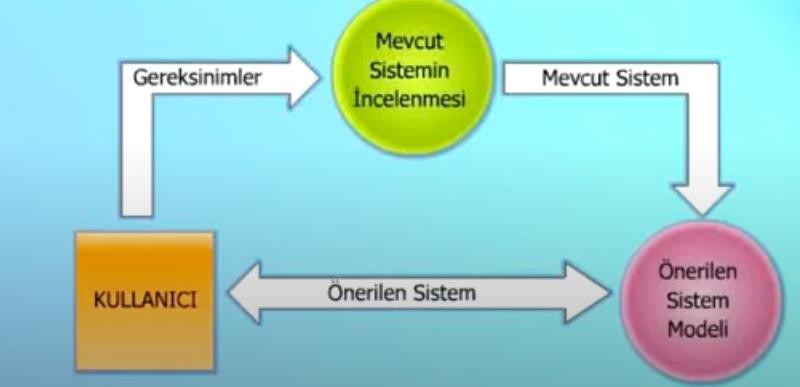
-Gereksinimlerin karşılandığını göstermek için sınama senaryoları oluşturulabilir mi? Ya da doğrudan sınama yapılabilir mi? Kullanıcı bir hız tanımlaması yapmadan ve yanıt süresinin

koşullarını tanımlamadan hızlı yanıt süresi isteyebilir. Örneğin hızlı yanıt süresi daha doğrulanabilir olarak şu şekilde tanımlanabilir; ‘sistemde kelime işlem programı kullanan en çok sayıda kullanıcı

ile sistem bir kullanıcının ekranına bir belgeyi bir sonraki sayfasıyla birlikte en çok 5 saniyede getirmelidir.

* 1. Gereksinimler izlenebilir mi?

-Her sistem işlevi kendisini zorunlu kılan bir dizi gereksinim olarak izlenebilir mi? Sistemim özel bir yönüyle ilgili bir dizi gereksinim bulmak kolay mı? Örneğin tüm iletişim gereksinimlerini gözden geçirmek için bütün gereksinimler okunmak zorunda mı?



MEVCUT SİSTEMİN İNCELENMESİ

-Bu çalışmada temel amaç yazılım geliştirilecek olan sistemin anlaşılması ve tanımasıdır. bu amaçla görüşme yapma gerekirse anket yapma yöntemleri kullanılır.

* Yapılacak görüşmelerde mevcut sistemde elle yürütülen işlemler girdi, işlev, çıktı ve diğer tüm işlevlerle olan ilişkiler bazında sorgulanır. İlgili yönerge korun ve yöntemler kullanıcıdan edinir.
* Sorgulama ya da inceleme işleminin belirtilmesi amacıyla çeşitli yöntemler kullanılır. sorgulama sonucu elde edilen duygularla ilgili olarak kullanıcıdan geri bildirim alınır.
* Mevcut sistemde elle yapılan işlerde kullanılan form defter ve yazma örnekleri sistemin boyutunu oluşturur. Bu tür bilgilerin birer kopyaları bir sonraki aşamada veri modellemesi için edinilir.

ÖNERİLEN SİSTEMİN MODELLENMESİ

-MANTIKSAL TASARIM-

-Mevcut sistemin modellenmesinden sonra bilgisayarla ortamda işlerin yapılabilmesi amacıyla önerilecek sistemi modeli oluşturulur.

* Bu modelleme daha çok bilgi sistemini geliştirecek teknik personele (sistem tasarımcıları, programlamacıları ) yöneliktir. Bu model aynı zamanda mantıksal model olarakta tanımlanır.
* Mantıksal model önerilen sistemin veri yapısını ve süreç yapısını hem genel hem de ayrıntılı olarak tanımlar. Mantıksal model kolaylıkla fiziksel modele (program parçaları veritabanı tabloları

vb.) dönüştürülebilir bir yapıdadır.

* Bazı yazarlar ‘mantıksal model’ üretimini tasarım aşamasının bir parçası olarak ele alırlar ve konuyu tasarım taslağı altında incelerler.

7

7.HAFTA : YAZILIM ÇÖZÜMLEME 2 GEREKSİNİM VERİSİ TOPLAMA:

Temel yöntemler;

1. Sorma
2. Psikolojik Türetme Yöntemi
3. İstatiksel Teknikler 1)Sorma Yöntemi:

* En eski ve en önemlilerinden biri.
* Gereksinim verilerinin toplanması için (İhtiyaçların doğru belirlenmesi için)
* Karşılıklı görüşme / anket a)Karşılıklı Görüşme:
  + Gereksinimlere ilişkin amaç , düşünce, yöntemler, duygu ve düşünceler araştırılır.
  + En etkin yollardan biri.

Piramit tarzı: Özel sorulardan genel sorulara doğru. Koni tarzı: Genel sorulardan özel sorulara doğru.

Elmas tarzı: Özel ile başlayıp genel sorularla sürer ve yine özel sorularla biter. b)Anket:

* Kullanıcı sayısı fazla ise
* Sistem üzerindeki aktörlerin fazla olduğu durumlarda
* Kullanıcı eğilimlerini ve davranış biçimlerini saptamak amacıyla
* Yazılı test (soru kısmı ve anket kısmı)

1. Psikolojik Türetme Teknikleri :

* Özellikle belirsizlik fazla ise
* Zayıf yapılı ortamlarda
* İnsan psikolojisine dayalı tekniklerle bilgi edinebilmek için
* Anket ve görüşmeye dayalı fakat onlardan farkı “Üçleme Tekniği” ni kullanmasıdır. (Psikolojide bir teknik)
* Algılama haritaları ve neden-etki çizelgeleri kullanmaktadır.

1. İstatistiksel Teknikler :

* Veri yoğunluklu ve veri hacminin yüksek olduğu ortamlarda verinin özelliklerini belirleme amacıyla
* Örnekleme yöntemi ve PIRA Modeli önemlidir.
  1. Örnekleme Yöntemi :

\*Amaç: veri toplama hızını arttırmak ve verilerdeki çelişkileri önlemektir.

* + Basit gelişigüzel örneklem, karmaşık gelişigüzel örneklem, amaçlı örneklem vb…
  + Örneklem boyutu belirli bir güvenilirlik düzeyinde olmak koşulu ile yine istatistiksel teknikler kullanılarak belirlenir.
  1. PIRA Modeli :
  + Personal, Interactive, Report and Analysis
  + Bilgi gereksinimlerinin tanımlarını belirli normlara bağlı olarak açıklamayı hedefler. Kişilerin bilgiye dayalı tercihlerini belirlemek amacıyla kullanılır.

VERİ MODELLEME YÖNTEMLERİ :

Sistemin mantıksal modelinde veri yapısını açıklamak amacıyla kullanılmaktadır. Veri yapısını çeşitli düzeylerde tanımlamayı (en soyuttan en ayrıntı düzeye) amaçlar.

En yaygın olanları :

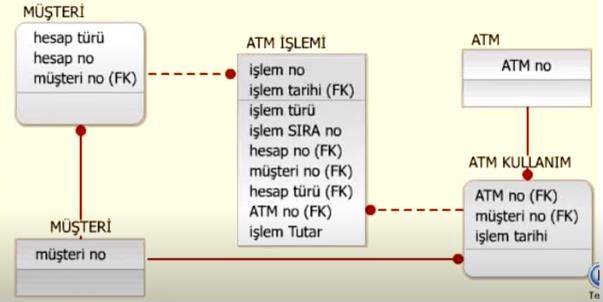
1. Nesne ilişki şemaları
2. Veri sözlüğü

1) Nesne İlişlki Şemaları :

* Bir veri nesnesi 3 temel özelliği ile bilinir:

1. Adı
2. Özellikleri
3. Diğer veri nesnesi varlıklarına referansı (Bu amaçla her bir veri nesnesini tek olarak belirleyen bir belirteç (anahtar) kullanılır. Söz konusu belirteç veri nesnesi varlığının ad özellikleri arasında yer alır.)

* Geliştirilecek sistemin kullancağı ana veri nesneleri ve aralarındaki ilişkiyi belirtir. Örnek :



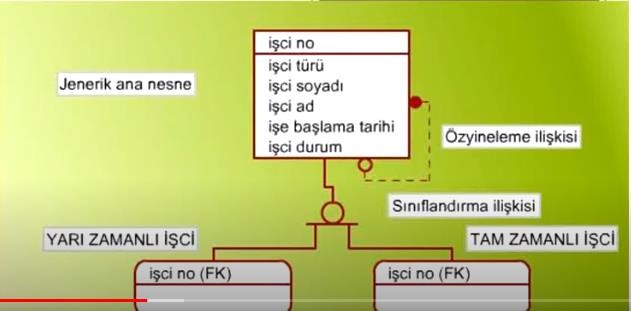
Burada örneğin ATM ve altında özellikleri verilmiştir ya da ATM işlemi ve ilişki içinde bulunduğu ATM kullanımı gibi nesnelerden dolayı foreign key olarak kendi içerisinde bu özelliği almıştır.



Örnek 2 açıklama : Javadan biliyoruz ki nesneler soyut olarak tanımladığımız sınıfların birer gerçek örneğidir. Somutlaştırma adına kullanıyoruz. Burada 2 nesne tanımlanmış biri insan nesnesi diğeri araba nesnesi. İnsan nesnesinin özellikleri olarak çeşitli nitelikler verilmiş (ad, adres, yaş, ehliyet no.. ). Aynı şekilde araba için de sıralanmış. Burada insan nesnesi le araba nesnesi arasında bir sahiplik ilişkisi vardır.

Örnek 3 açıklama : Burada artık somut olarak bir veritabanı tablosu mevcut. Dolayısıyla biz nesne ilişkisi diyagramlarında kendi veritabanı tablolarımızı oluşturuyoruz.





Yine varlık ilişki diyagramlarındakine benzer olarak burada nesneler arasındaki ilişkileri birden bire, birden çoka ve çoktan çoka şeklinde tanımlayabiliyoruz.

1-1 ilişki : Bir insan ancak 1 araba sahibi olabilir.

1-N ilişki : Bir insan birden çok araba sahibi olabilir.

M-N ilişki : Birden insan birden çok araba sahibi olabilir. VERİ SÖZLÜĞÜ:

Nesne ilişki şemalarında belirtilen nesne özelliklerinin ayrıntılı tanımları veri sözlüğünde yer alır.

* Veri adı,
* Veri eş adı (Aynı veri için kullanılan diğer ad)
* Nerede / nasıl kullanıldığı
* İçerik tanımı

SÜREÇ-İŞLEM MODELLEME YÖNTEMLERİ:

Geliştirilecek sistemin süreç ya da işlemlerini ve bu süreçler arasındaki ilişkileri tanımlamak amacıyla kullanılan yöntemlerdir.

* Veri akış diyagramları (VAD)
* Süreç tanımlama dili (STD)
* Karar tabloları
* Karar ağaçları
* Nesne şemaları

1. Veri Akış Diyagramları (VAD):

* Sistemin mantıksal modeli yukarıdan aşağıya bir yaklaşımla oluşturulur
* Önce en genel biçimiyle (Yalnızca dışsal ilişkiler) sonra iç yapısındaki süreç ve ilişkiler modellenir.

VAD KÜMESİ :

3 tür veri akış diyagramı bulunuyor:

1. Kapsam Diyagramı :

* Dışsal ilişkilerini göstermek amacıyla kullanılır.
* Yalnızca bir çember ve gerekli sayıda veri kaynağı ile aralarındaki veri akışlarının belirtiminden oluşur.

1. Genel Bakış Diyagramı:

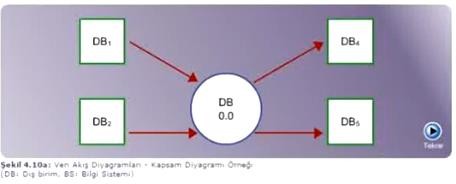
* Kapsam diyagramının ayrıştırılmış biçimidir.
* Bilgi sisteminin ana işlevlerini, bu işlevlere ilişkin veri kaynaklarını , veri depolarını ve işlemler arasındaki ilişkileri içeren VAD.
* En çok birkaç tane çizilir.

1. Ayrıntı Diyagramı :

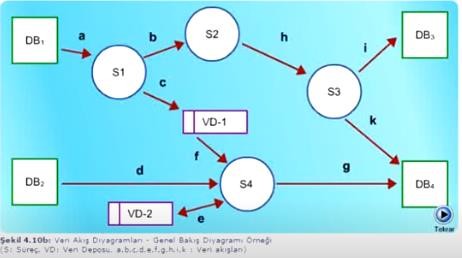
* Genel bakış diyagramında belirtilen her ana işlev, belirli bir ayrıntı düzeyine kadar (en fazla 7+/-2) detay VAD oluşturularak ayrıştırılır.
* Genellikle detay VADlarda veri kaynakları gösterilmez.

VAD Neyi Gösterir?

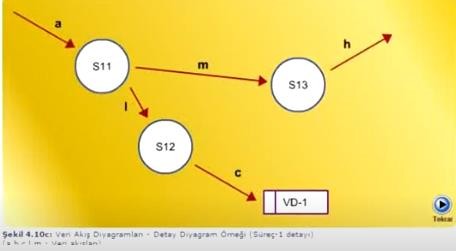
* Sistemin durağan yapısını
* Süreçlerini ve aralarındaki veri akış ilişkilerini
* Kurum birimlerini ya da dış birimleri bilgi kaynakları olarak gösterir.
* Gerekli olan ana veri depolarının neler olduğunu ve hangi süreçler tarafından kullanıldığını
* Süreçleri yukarıdan aşağı doğru ayrıştırarak gösterir. (En genelden en ayrıntı düzeye kadar.) VAD Neyi Göstermez?
* Zamana ilişkin durumunu ve bu durumla ilgili bilgileri göstermez.
* Bilgi sistemi süreçlerinin kendi aralarındaki “karar” ilişkisini göstermez.
* Gerek bilgi sistemi süreçleri gerek veri akışları gerekse bilgi kaynakları ve bilgi depoları için ayrıntı içermez.



Örneğin burada bir kapsam diyagramı mevcut. DB dediğimiz bir süreç var ve bu süreçlere bağlı veri kaynaklarımız mevcut. Burada DB herhangi bir dış birimi gösteren bir ifade olarak özellik almış. BS dediğimiz yapılar ise bilgi sistemi olarak özetleniyor. DB1 ve DB2 den bir veri akışı var ve bu veri akışıyla birim dediğimiz yapımız DB4 ve DB5 i verilerle besliyor.



Genel bakış diyagramı kapsam diyagramlarından biraz daha ayrıntılanmış bir yapı. Burada S dediğimiz yapı süreçtir. VD, veri deposu ve DB ler ise dış birim. Burada yapı biraz daha dağıtılarak süreçler ortaya konmuş.

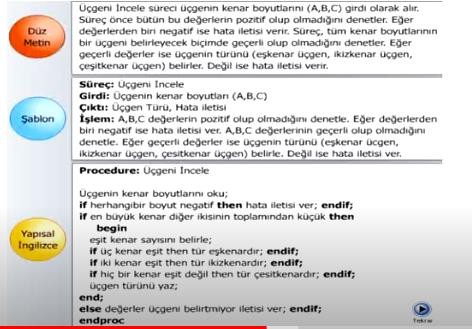


Süreç1 in detayını veren detay diyagram örneği

1. Süreç Tanımlama Dili :

Veri akış diyagramlarında ayrıştırılmış olan bilgi sistemi süreçlerinin iç yapılarını belirtmek amacıyla araç, yöntem ya da gösterim biçimleri :

* Düz metin,
* Şablon,
* Yapısal İngilizce (pseudo koda benzer) Örnek:



KULLANICI ARAYÜZ PROTOTİPLEME (KAP) :

* Gereksinim tanımlama ya da sistem çözümleme çalışmasının önemli bir bileşeni, oluşturulacak bilgi sistemine ilişkin girdi ve çıktı gereksinimlerinin tanımlanmasıdır.
* Gereksinim çalışmasının hemen sonunda kullanıcıya gösterilecek bir prototip yazılım

hazırlanmasını içerir. Söz konusu prototipin gerçekten içsel olarak çalışmayan ancak ekranlar, menüler ve bunların aralarındaki geçişlerin çalıştığı bir yazılımdır.

KAP İçin Ekranlar Nasıl Hazırlanır?

* Hazırlanması için gerekecek iş gücü ve zaman, sistem çözümleme için ayrılan zamanın

%5ini aşmamalıdır. (Kod olmaksızın sadece panel, menü vs bileşenlerle hazırlanıyor.)

* Bir özellik birden fazla ekranda yer alsa bile yalnızca bir kez gösterilmelidir.
* Hiçbir kişisel ve içsel işlem içermemelidir.

Örneğin; bir ekranda kişilerle ilgili sorgulama yapıldığını düşünelim. Adı ve soyadı bilinen bir kişinin, bu bilgi verildiğinde, diğer ayrıntı bilgilerinin ekranda görüntüleneceğini varsayalım. Kullanıcı, ad soyad bilgisini girecek ve ilgili sorgulama tuşunu etkinleştirdikten sonra ekranda hep aynı bilgler görünecektir. Yani KAP, Ad soyad bilgisine göre kişi tablosundan arama yapmayacak, program içerisinde hep aynı sonuçları ekrana verecetir. Çünkü kişi tablosunun yaratılması ya da kullanılması söz konusu değildir. Burada amaç, kullanıcıya sorgulama biçimini ve sorgulama sonucunda elde edilecek bilgileri açıklamaktır. Sorgulanan bilginin

içeriği bu aşamada önemli değildir. Kullanıcıdan sorgulama sonucu gelen bilgi türlerine ilişkin, sorgulama işleminin işlem akışına ilişkin bilgi alınması hedeflenir.

KAP İçin Raporlar Nasıl Hazırlanır?

* Yazıcı çıktısı biçiminde alınması istenen raporlar bir metin düzenleyici (örneğin MC Word ) aracılığıyla hazırlanır ve belirli bir biçimde numaralandırılır.
* Örneğin; bilgi sisteminin insan kaynakları yönetim bilgi sistemi (IKYS) olduğunu

varsayalım. Raporları MS Word ile hazırlayıp bunlara “IKYS-AT-R1” , “IKYS-AT-R2” gibi numaralar verelim. Kullanıcıya prototip gösterdiğimizde kullanıcı, rapor alma düğmesini

etkinleştirdiğinde “Şu anda yazıcıdan IKYS-AT-R1 nolu rapor dökülmekte.” şeklinde bir ileti alacaktır. Bu sayede kullanıcı bize rapor alanları, raporun tetkiklendiği ekranla ilgili geri bildirimlerinive değişiklik istediğini iletir. Bu değişiklikler yapıldıktan sonra söz konusu raporla ilgili gereksinimler kesinleşmiş olur.

Çözümleme Çalışması Nasıl Değerlendirilir?

Sistem çözümleme çalışması sonuçlandırıldıktan sonra elde edilen ara ürünün (mantıksal model) istenenleri karşılayıp karşılayamadığının belirlenmesi amacıyla değerlendirilir.

Temel olarak ;

\*Kullanıcıların benimsemesi ve onayı

* Tutarlılık, tamlık, uygunluk
* Tasarım çalışması için yaterliliği
  1. Tamlık ve Tutarlılık :

Tamlık : Tüm ögeler ve bunlar arasındaki ilişkilerin tanımlanmasını gerektirir. Tutarlılık : Modelin kendi içerisinde hatasız çelişkisiz olmasıdır.

Tamlık ve tutarlılık denetimi, bir programlama dilinde yazılmış bir programın söz dizim kurallarının denetimine benzer.

Örneğin; modelleme aracı olarak VAD kullanıldığında

* Bütün süreçlerin girdi ve çıktıları var mı?
* Süreçler, veri akışları, dış birim ve veri depoları uygun şekilde adlandırılıp numaralandırılmış mı?
* VAD deki tüm isimler veri sözlüğünde yer almış mı?
* Planlama aşamasında öngörülen belirtimlerin hepsi ele alınmış mı?
  1. Olurluluk:

\*Sistem çözümleme aşamasındaki çalışmanın planlama aşamasında yapılan çalışmaya uygunluğunun belirlenmesi için yapılan çalışmaları içerir.

* Maliyet kestirim çalışması yinelenir, sapmalar saptanır, kaynaklar yeniden planlanır. Sapmalar fazla olursa çalışmanın yeniden gözden geçirilmesi gerekir.
* Bu aşamada maliyet kestirimi için daha fazla bilgi mevcuttur. VAD yöntemi, maliyet kestirim yöntemlerinin kolayca uygulanabilmesini olanaklı kılar.

VAD’den Elde Edilenler:

Girdi ve çıktı sayısı: Doğrudan genel bakış diyagramındaki veri akışlarının incelenmesiyle elde edilir.

Kütük Sayısı: Genel bakış diyagramlarındaki veri depoları sayılarak elde edilir.

Sorgu Sayısı: Genel bakış diyagramı ve detay diyagramlarındaki süreçlerin incelenmesiyle elde edilir.

Arayüz Sayısı: Kapsam diyagramındaki dış birimlerle olan ilişkiler incelenerek elde edilir.

ALINAN DERSLER ÖZET :

* Kullanıcılar, genellikle görüşme tutanaklarını imzalamakta çekingenlik gösterirler.'Siz hazırlayın, bize gönderin bizde imzalar size göndeririz ' biçiminde yöntemler önerirler.
* Görüşme tutanaklarınızı anında iki kopya olarak tutun ve hemen görüşme bitiminde

imzalatma yöntemini benimseyin ve uygulayın. Aksi durumda, ileride ortaya çıkabilecek sorunlarda kaybeden taraf siz olursunuz.

* Tüm görüşme ve toplantı kayıtlarınızı olabildiğince bilgisayarlı ortamda saklayın. Aradan, uzun bir süre geçtikten sonra, eski kayıtlara erişimde büyük kolaylıklar sağladığını göreceksiniz.

Bu amaçla Visual Source Safe türü basit bir araç bile kullanabilirsiniz.

* VAD yöntemi, kullanıcı tarafından algılanma açısından oldukça etkili bir yöntemdir. Bu durum, değişik projelerde görülmüştür.
* Kullanıcıya, görüşme ile ilgili olarak düzeylendirilmemiş bir VAD hazırlayıp

gönderdiğinizde, kullanıcının bu VAD'nı kendi başına yeniden çizebildiği gözlemlenmiştir.

* CASE aracı kullanımının modelleme çalışmasını oldukça hızalandırdığı gözlemlenmiştir. Niteliği ne olursa olsun mutlaka bir CASE aracı kullanın.

CASE aracı kullanımı ayrıca, daha sonraki aşamalarda oluşabilecek ve mantıksal modele yansıtılması gereken günlemelerin kolayca yapılmasını sağlar.

8

TASARIM

Bu aşamada yapısal sistem geliştirme metodolojisine göre yapılan bir çözümlemenin tasarıma dönüştürülmesi, tasarım kavramları, veri tasarım ve süreç tasarım yöntemleri açıklanmaktadır.

Tasarım sistem analizi çalışması sonucu üretilen mantıksal modelin fiziksel modele dönüştürülmesi çabasıdır.

Fiziksel model gerektirecek yazılımın;

Hangi parçalardan oluşacağını,bu parçalar arasındaki ilişkilerin nasıl olacağını parçaların iç yapısının ayrıntılarını,gerekecek veri yapısının fiziksel değişimini(dosya,veri tabanı, Hash tablosu, vektör..vs)

tasarımını içerir.

Tasarım Kavramları

Soyutlama (Abstraction):Detayları gizleyerek yukardan bakabilme imkanı sağlar (nesne tabanlı programlama paradigmalarından da biridir.)

İyileştirme(enhancement):Soyutlama düzeyinde irdeleme bittikten sonra,daha alt seviyelere inerek tanımlamalarda ayrıntı, bazen de düzeltme yapılarak tasarımın daha iyi kesinlik kazanması sağlanır.

Modülerlik(modularity)[en önemlisi]:Sistemi istenilen kalite faktörleri ışığında parçalara ayrıştırma sonucu elde edilir

Bütün karmaşıklığın tek bir modülde toplanması yerine,anlaşılabilir ve dolayısıyla projenin zihinsel kontrol altında tutulması için sistem birçok modüle ayrılır.[Hem yönetim olarak kolaylık sağlar hem yazılım karmaşıklığını ortadan kaldırır.]

Modüller, isimleri olan tanımlanmış işlevleri bulunan ve hedef sistemi gerçekleştirmek üzere tümleştirilen birimlerdir.

Sistem modüllerinde dallanan, ağaç veri modülü gibi düşünülebilir. Sistem dallanarak farklı alt modüllere ayrılır.

İşlevsel Bağımsızlık

Modüllere parametreyle değer gönderilir ve sonuç değer alınır.Bu modülü çağıran program parçası sadece bu sonucu kullanabilir.Çağrılan modülün işlevsel olarak yaptıklarıyla ilgili değildir.

Veri Tasarımı

Yapı tasarımı,arayüz tasarımı ve süreç tasarımından önce yapılması gereken ilk tasarım veri tasarımıdır.

Bilgi saklama ve soyutlama bu işlem için önemli kavramlardır.

Yaparken dikkat edilecekler:

Değişik veri yapıları değerlendirilmelidir.

Bütün veri yapıları ve bunlar üzerine yapılacak işlemler tanımlanmalıdır. Alt düzeyde tasarım kararları tasarım süreci içerisinde geciktirilmelidir.

Bazı çok kullanılan veri yapıları için bir kütüphane oluşturulmalıdır. Kullanılan programlama dili soyut veri tiplerini desteklemelidir.

Yapısal Tasarım

Ana hedefi modüler bir yapı geliştirip modüller arasındaki kontrol ilişkisini temsil etmektir.Ayrıca yapısal tasarım bazen de veri akışını gösteren biçime dönüştürülebilir.

Veri akışları üç parçada incelenebilir. Girdi akışı, Çıktı akışı, İşlem akışı

Ayrıntı Tasarım-Süreç Tasarım

Süreç tasarımı veri, yapı ve arayüz tasarımından sonra yapılır.

İdeal şartlarda bütün algoritmik detayın belirtilmesi amaçlanır.

Ayrıca süreç belirtiminin tek anlamı olması gerekir, değişik şahıslar tarafından farklı yorumlanmamalıdır.

Doğal diller kullanılabilir. Süreç tanımlama dili(PDL) Yapısal program yapıları

Yapısal programıın temel amacı program karmaşıklığını en aza indirmek ve anlaşılabilirliğini artırmaktır.

Bu amaçla şu yapılar kullanılır:

Ardışıl işlem yapısı Koşullu işlem yapısı Döngü yapısı

(GOTO (dallanma komutu) kullanımı uygun değildir.)

Ardışıl işlem Şartlı işlem Tekrarlı işlem

Sayı<0

işlem

Sayı>=0

Sayı=sayı-1

Sonuç göster

2’yle çarp

Sayı oku

Karar tabloları

Karmaşık koşul değerlendirmeleri yapmak gerekir. Bunların düzenli bir gösterimi karar tablolarında yapılabilir.



Tasarımda Olması Gereken Ortak 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 kullanabileceği ve kullanamayacakları özellikleri ifade eder.

İşlem bazında yetkilendirme Ekran bazında yetkilendirme

Ekran alanları bazında yetkilendirme Güvenlik alt sistemi

Yapılan bir işlemde, işlemi yapan kullanıcının izlerinin saklanması işlemleri LOG Filer (sistem günlüğü)

Yedekleme alt sistemi

Her bilgi sisteminin olağandışı durumlara karşı yedekleme sistemi olmalıdır. Veri iletişim alt sistemi

Coğrafi olarak dağıtılmış hizmet birimlerinde çalışan makineler arasında veri akışının sağlanması işlemleri

Çevrimiçi veri iletimi(real-time)

Çevrimdışı veri iletimi(disketler,teypler) Arşiv alt sistemi

Belirli bir süre sonrasında sık olarak kullanılmayacak olan bilgilerin ayrılması ve gerektiğinde bu bilgilere erişimi sağlayan alt sistemleridir.

Aktif veri sistemi Dönüştürme alt sistemi

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

Kullanıcı Arayüz Tasarımı

Kullanıcıyla ilişkisi olmayan arayüzler: Modüller arası arayüz

Sistemle dış nesneler arası arayüz Kullanıcı arayüzleri:

Kullanım kolaylığı ve öğrenim zamanı esastır Program=arayüz yaklaşımı vardır

Genel prensipler:

Veri giriş formlarının tutarlı olması Önemli silmelerde teyit alınması Yapılan çoğu işlem geri alınabilmesi

Hataların affedilmesi yanlış girişte kırılmama Komut isimlerinin kısa ve basit olması

Menülerin ve diğer etkileşimli araçların standart yapıda kullanımı Bilgi Gösterimi:

Konu çerçevesinde bilgi gösterilmeli. Kullanıcı bunaltılmamalı. Grafik ve resim kullanılmalı. Hata mesajları açıklayıcı ve anlaşılır olmalı. Değişik tür bilgiler kendi içinde sınıflandırılmalı. Rakamsal ifadelerle analog görüntü verilmeli(%89 değil).

Veri girişi

Kullanıcı hareketleri en aza indirilmeli

Gösterim ve girdi sahaları birbirinden ayrılmalı (renk uyumu önemli)

Kullanıcı uyarlamasına izin verilmeli, kullanıcı bazı özellikleri tanımlayabilmeli Kullanılan konuyla ilgili gereksiz komutlar deaktifleştirilmeli

Bütün girdiler için yardım kolaylıkları olmalı Kullanıcı Arayüz prototipi

Tasarım çalışması sonucunda daha önceden gereksinim çalışması sırasında hazırlanmış olan kullanıcı arayüz prototipi,ekran ve rapor tasarımları biçimine dönüşür. Ekranlar son halini alır, raporlar

kesinleşir. Kullanıcıya gösterilerek onay alınır .

Tüm programının tek elden çıktığının ifade edilebilmesi açısından tüm ekranların aynı şablon üzerine oturtulması önemlidir.

Menü çubuğu Gövde (değişebilir) Araç çubuğu Durum çubuğu

Başlangıç Tasarım Gözden Geçirme

Analiz aşamasında belirlenen gereksinimleri karşılayıp karşılanmadığının belirlenmesidir.

Sistem gereksinimlerine yardımcı olan kullanıcılar, sistem analizi yapan çözümleyiciler, sistemin

kullanıcıları, tasarımcılar, yönlendirici, sekreter, sistemi geliştiricek programcılar’dan oluşan bir grup tarafından yapılır.

Ayrıntılı Tasarım Gözden Geçirme

İlk aşama tamamlandıktan sonra tasarımın teknik uygunluğunu belirlemek için yapılır. Bu çalışmada; Çözümleyiciler , Sistem tasarımcıları, Sistem geliştiriciler, Sekreter’den oluşan bir ekip kullanılır.

Kalite Tasarım Ölçütleri Bağlaşım (coupling):

Tasarımı oluşturan modüller arası ilişki ile ilgilidir. Yapışıklık (cohesion):

Modülleerin iç yapısı ile ilgilidir.

Bağlaşım

Modüller arası bağlılığın ölçülmesi için kullanılan bir ölçüttür. Yüksek kaliteli bir tasarımda bağlaşım ölçümü az olmalıdır yani her bir modül olabildiğince farklı işler yapacak şekilde tasarlanmalı ve diğer modüllerle olabildiğince az etkileşimi olmalıdır.

Bağlaşımın düşük olması;

Hatanın dalgasal yayılma özelliğinin azalması, modüllerin bakım kolaylığı ,modüller arası ilişkilerde karmaşıklığın azaltılması nedenleriyle istenmektedir.

Bazı türleri vardır:

Yalın Veri Bağlaşımı

Herhangi iki modül arası iletişim yalın veriler (tamsayı,karakter,boolean vs.)aracılığla gerçekleştiriliyorsa bu iki modül yalın veri bağlaşımlıdır.

Karmaşık Veri Bağlaşımı

Herhangi iki modül arası iletişimde kullanılan parametreler karışık veri yapısı(kayıt,dizi,nesne vs.)olması durumunda tanımlanır.

Denetim Bağlaşımı

Herhangi iki modül arası iletişim parametresi olarak denetim verisi kullanılıyorsa olur. Ortak Veri Bağlaşımı

Eğer iki modül ortak bir alanda tanımlanmış verilere ulaşabiliyorsa bu iki modül ortak veri bağlaşımlı olarak tanımlanır.

Fazla istenmeme sebepleri; ortak veri alanını izlemek zordur, ortak veri kullanan modüllerde yapılan değişiklikler diğer modülleri etkiler, ortak veri üzerinde yapılacak değişikliklerde bu veriyi kullanacak bütün modüller göz önüne alınmalıdır.

İçerik Bağlaşımı

Modüllerin iç içe tasarlanması sonucu, bir modülün başka bir modül içerisinde tanımlanmış veri alanına erişebilmesi olanaklaşır ve bu durum içerik bağlaşımına yol açar.

Yapışıklık

Bağlaşım istenmeyen bir özellikken yapışıklık istenen bir özelliktir. İkisi ters orantılıdır.

Bir modülün kendi içindeki işlemler arasındaki ilişkilere ilişkin bir ölçüttür. Modül gücü olarak da tanımlanır.

İşlevsel Yapışıklık

Tek bir iş problemine ilişkin sorunu çözen modül olarak tanımalanır. Örneğin maas\_Hesapla adlı bir methodumuz modülümüz maaş hesaplama dışında bir iş yapmaz.

Sırasal Yapışaklık

Bir modülün içindeki işlemler incelendiğinde, bir işlemin çıktısı, diğer bir işlemin girdisi olarak kullanıyorsa bu modüldür.

Ham\_veri\_kaydını\_düzelt

Düzeltilmiş\_ham\_veri\_kaydını\_doğrula Doğrulanmış\_kaydı\_gönder

İletişimsel Yapışıklık

Bir modülün içindeki farklı işlemler aynı girdi ya da çıktıyı kullanıyorlarsa bu modüldür. Sicil\_no\_yu\_al

Adres\_bilgilerini\_bul Telefon\_bilgisini\_bul Yordamsal Yapışıklık

Bu modüldeki işlemler arasında denetim ilişkisi bulunmaktadır. İşlemlerin birbiriyle veri ilişkisi yoktur,ancak işlem sırası önemlidir.

Ekran\_görüntüsü\_yaz Giriş\_kaydını\_oku Zamansal Yapışıklık

Modül içerisindeki işlemlerin belli bir zaman içerisinde uygulanması gerekiyor fakat bu işlemlerin birbirleriyle bir ilişkisi yok, yani işlem sırası önemli değilse zamansal yapışıklık vardır.

Kapıyı\_Aç Kamerayı\_Çalıştır Mantıksal Yapışıklık

Mantıksal olarak aynı türdeki işlemlerin bir araya toplandığı modüller mantıksal yapışık olarak adlandırılır. Örnek: dizilere değer atama işlemleri

Gelişigüzel yapışıklık

İşlemler arasında herhangi bir ilişki bulunmaz. Ara\_kayıt\_oku

Stok\_kutuğu\_oku Hata\_iletisi\_yaz

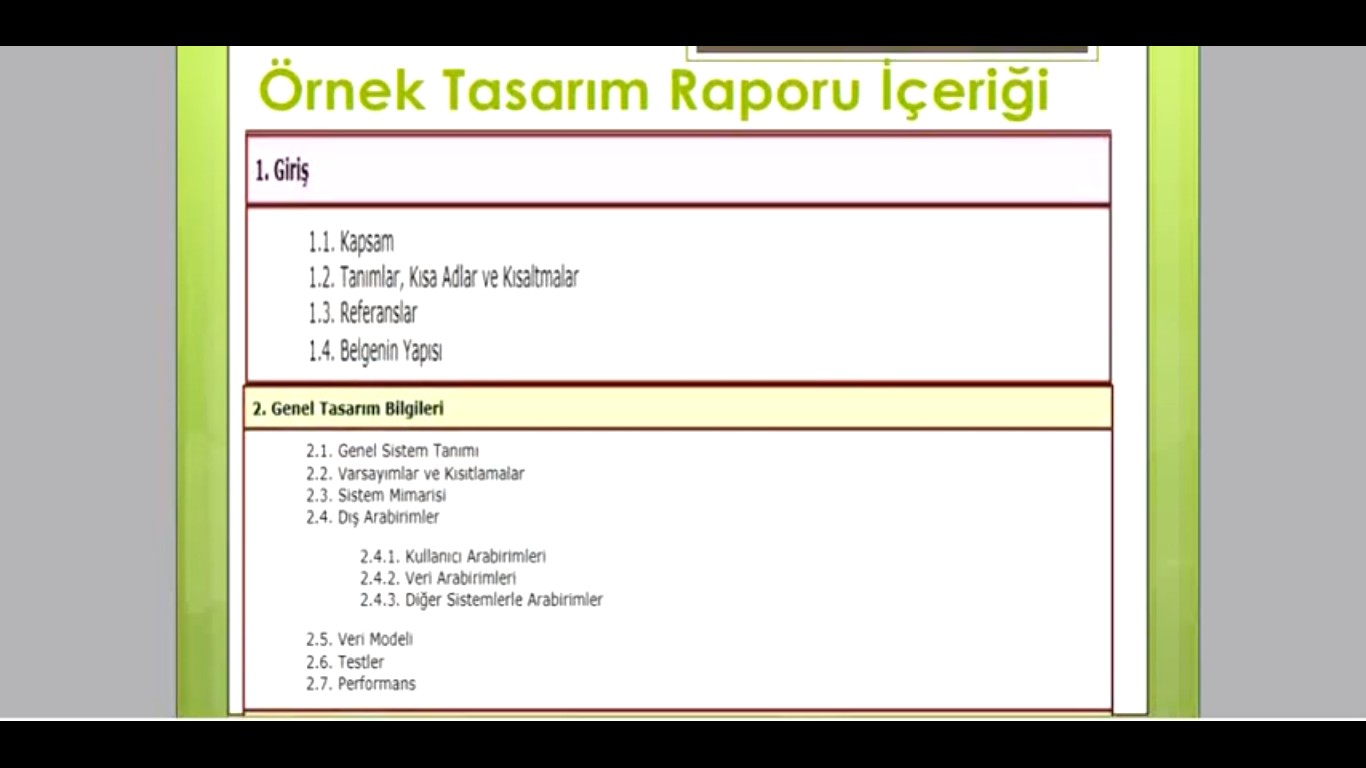
Yapışıklık hesaplama

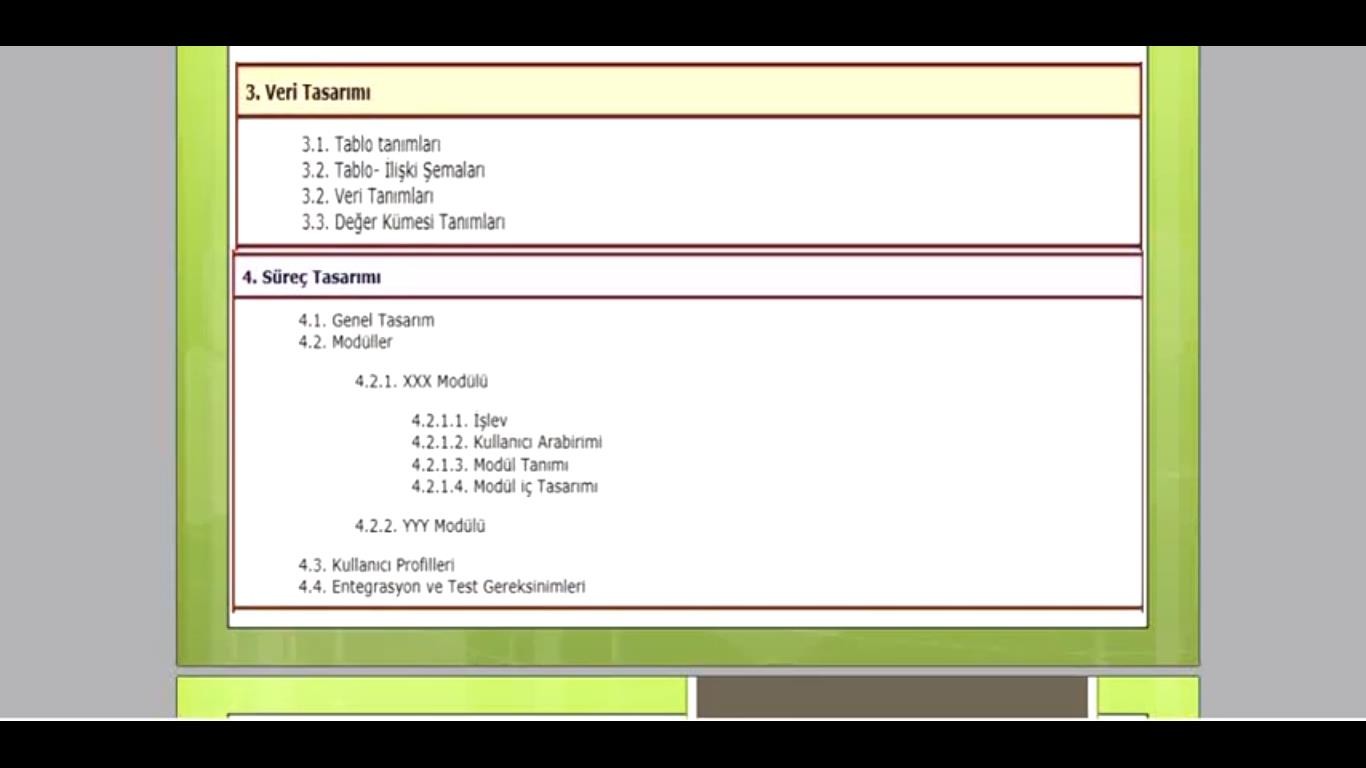
Genel olarak bir modülün yapışıklık düzeyini hesaplamada bazı güçlüklerle karşılaşılabilir. Çünkü bir modüldeki işlemler hep aynı tür özellik göstermeyebilir.

|  |  |  |
| --- | --- | --- |
| Yapışıklık türü | Ölçek | Kalite |
| İşlevsel | 1 | En iyi |
| Sırasal | 2 |  |
| İletişimsel | 3 |  |
| Yordamsal | 4 |  |
| Zamansal | 5 |  |
| Mantıksal | 6 |  |
| Gelişigüzel | 7 | En kötü |



Örnek Tasarım Raporu İçeriği:





Yazılım Gerçekleştirm

### Gerçekleştirme Fazı:

1. Programlama Dilleri
2. Yaılım Gerçekleştirme
3. Derleyiciler
4. Kodlama Etkinliği ve Kuralları
5. Modüler Gerçekleştirme
6. Gerçekleştirmenin Belgelendirilmesi

### Yazılım Geliştirme Ortamlarında bulunan unsurlar:

* Programlama Dili
* Veri Tabanı Yönetim Sistemi
* Hazır Program Kitapçıkları(Kütüphane dosyaları)
* CASE Araçları

# a) Programlama Dilleri:

Geliştirilecek uygulamaya göre kullanılabilecek programlama dili de değişmektedir.

**\*Uygulama alanlarının sınıflaması:**

-Veri işleme yoğunlamalar

-Hesaplama yoğunluklu uygulamalar

-Süreç ağırlıklı uygulamalar

-Sistem programlamaya yönelik uygulamalar

-Yapay (zeka-akıl) uygulamaları biçiminde verilmektedir.

### Veritabanı Yaklaşımının Geleneksel Kütük Kavramından Ayıran Özellikleri:

1. Veri sözlüğü. 2- Veri soyutlama. 3- Program-Veri ve program -işlem bağımsızlığı 4- birden çok kullanıcı desteği 5- Verinin birden fazla işlem arasında paylaşımı
   * **Veri Soyutlama:**

Veri tabanı yaklaşımının bir temel karakteristiği kullanıcıdan verinin saklanması Konusundaki detayları gizleyerek veri soyutlamasını sağlamasıdır. Bu soyutlama Sağlayan ana araç veri modelidir.

## Program-veri ve program-işlem bağımsızlığı:

-Geleneksel dosya işleme yönteminde,veri dosyalarının yapısı bu Dosyalara erişim programları içine görmülüştür.

-Tersine VTYS erişim programları veri dosyalarından bağımsız olarak Tasarlanmışlardır. VTYS de saklanan veri dosyalarının yapısı erişim Programlarından ayrı olarak katalogda tutulduğundan program

ve veri bağımsızlığı sağlar.

# Veri Modelleri:

**Kavramsal veri**

**modelleri**

**Gösterimsel veri modelleri**

**Yüksek Düzey modelleri**

**Fiziksel veri modelleri**

* **Şemalar:**
  + Herhangi bir veri modelinde veri tabanının tanımlanması ile kendisini ayırmak öneml Veri tabanının tanımlamaları veri tabanı şeması veya meta-veri olarak adlanılır.

Veri tabanı şenası,tasarım sırasında belirtilir ve sıkça değişmesi beklenmez.

* + Pek çok veri modeli şemaları,diyagramlar halinde göstermek için belli gösterim Biçimlerine sahiptir. Diyagramlar her kayıt tipinin yapısını gösterir fakat kaydın gerçek Örneğini göstermez.

### CASE Araç ve Ortamları:

* + CASE Araçları, yazılım üretiminin hemen her aşamasında ( Planlama – Çözğmleme – Tasarım – Gerçekleştirme – sınama ) üretilen bilgi ya da belgelerin bilgisayar ortamında Saklanmasını, bu yolla kolay erişilebilir ve yönetilebilir olmasını olanaklı kalır.
* **Kodlama stili:**
  + Yazılım kod stili konusunda herhangi bir kabul edilmiş standart bulunmamaktadır.
  + Bu konuda, yazılım geliştirilen ekiplerin, kodlama aşamasına başkamadan kodların Düzeni konusundaki standartlarını ya da kurallarını geliştirmeleri ve bu kuralların

Uygulamaya geçmesini sağlamaları önerilmektedir.

### McCabe Karmaşıklık Ölçütü Hesaplama

Ana program ve alt programlar çizge biçimine dönüştürüldükten sonra programın McCabe karmaşıklık öiçütü V(G) aşağıdaki biçimde hesaplanır.

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

K : kenar sayısı D : düğüm sayısı P : bileşen sayısı

## Gerçekleştirme Öznitelikleri:

-Etkinlik - Belirgin Arayüz

-Aykırı Durumların - Taşınabilirlik

Kotarılması -Çok düzeyli koruma

-Hata Ayıklama -Gözden geçirme

-Atık/Çöp toplama -Belgelendirme

-Soyutlama -Bilgi Gizleme -Etiketleme -Güvenlik -Basitlik

YAZILIM DOĞRULAMA VE GEÇERLEME

-Doğrulama

Doğru ürünümü üretiyoruz?

=Ürünü kullanacak kişilerin isteklerinin karşılanıp karşılanmadığına dair etkinliklerden oluşur

-Gerçekleme

Ürünü doğru olarak mı üretiyoruz?

Ürünün içsel niteliğine ilişkin izleme ve denetim etkinliklerinden oluşur.

IEEE 1028 standardı; hem kritik hem de kritik olmayan yazılımlar için geçerli olan

gözden geçirme ve denetleme için tanımlar ve yeknesak prosedürler sağlanır. Süreçler, yönetim incelemesi, teknik inceleme, yazılım incelemesi, izlenecek yol ve denetimi

içerir. Tanımlanan süreçler, yazılım yaşam döngüsü boyunca uygulanabilir. SVVP (Software Verification and Validation Plan)

Yazılım proje yönetimi, yazılım testi ve yazılım mühendisliğinde doğrulama ve onaylama ( **V&V** ), bir yazılım sisteminin amaçlanan amacını yerine getirmesi için spesifikasyonları ve **gereksinimleri** karşılayıp karşılamadığını kontrol etme sürecidir . Yazılım kalite kontrolü olarak da adlandırılabilir.

Küçükten büyüğe sınama kavramlar

1. Birim sınaması Bağlı oldukları diğer sistem unsurlarından bütünüyle soyutlanmış olarak birimlerin doğru çalışmalarının belirlenmesi amacıyla yapılır.
2. Alt-Sistem sınama modüllerin bütünleşmesi ile oraya çıkar. Bu düzeyde en çok hata ara yüzlerde bulunmaktadır.
3. Sistem Sınaması Üst düzeyde bileşenlerin sistem ile olan etkileşimlerinde çıkacak hataları arar.
4. Kabul sınama sistemin çalıştırılmadan önce ki son sınamasıdır. Artık yapay veriler yerine gerçek veriler kullanılır.

SINAMA YÖNTEMLERİ

Her mühendislik ürünü, iki yoldan biri ile sınanır: Sistemin tümüne yönelik işlevlerin doğru yürütüldüğünün (kara kutu – Black box) veya iç işlemlerin belirtimlere uygun olarak yürütüldüğünün bileşenler tabanında sınanması (beyaz kutu – White box).

Kara kutu sınamasında sisteme, iç yapısı bilinmeksizin gelişigüzel girdiler verilerek sınama yapılır.

Beyaz kutu sınaması tasarlanırken, birimin süreç belirtiminden yararlanılır.

Bütün bağımsız yollar en az bir kere sınanır, bütün döngüler sınır değerlerinde sınanır ve iç veri yapıları denenir.

Not: Kara kutu sınamasında kod sınamaları olmazken beyaz kutu sınamalarında mantıksal karar yapılarına kadar sınama yapılır. Çoğu zaman, kullanılma olasılığı çok az olarak kestirilen program yolları, aslında çok sıkça çalıştırılıyor olacaktır. Yazım hataları rastgele olarak dağılır. Bunlardan bazılarını derleyiciler bulur, bazıları da bulunmadan kalır.

TEMEL YOLLAR SINAMASI

Bağımsız yolların saptanması için program ilk önce graf (çizgisel) biçime çevrilir. Bunu yapmak için ise, program iş akış şemaları oldukça iyi bir başlangıç noktasıdır.

SINAMA VE BÜTÜNLEŞTİRME STRATEJİLERİ

Örneğin, yukarıdan aşağı ve aşağıdan yukarı stratejileri bütünleştirme yöntemine bağımlıdır. Ancak işlem yolu ve gerilim sınamaları, sistemin olaylar karşısında değişik işlem sıralandırmaları sonucunda ulaşacağı sonuçların doğruluğunu ve normal şartların üstünde zorlandığında dayanıklılık sınırını ortaya koyar. Yukarıdan aşağı bütünleştirmede önce en üst noktadaki bileşen, bir birim/modül/alt sistem olarak sınandıktan sonra alt düzeye geçilmelidir.

Alt bileşenler ise bu stratejiye göre henüz hazırlanmış olmadıklarından bunların yerine üst bileşenin sınaması için kullanılmak üzere koçan programlar yazılır. Koçanlar, bir alt bileşenin, üst bileşen ile arayüzünü temin eden fakat işlevsel olarak bir şey yapmayan boş çerçeve programlardır.

YUKARIDAN AŞAĞI SINAMA ve BÜTÜNLEŞTİRME

1-Düzey Öncelikli Bütünleştirme

En üst düzeyden başlanır, öncelikle aynı düzeylerdeki birimler bütünleştirilir. 2-Derinlik Öncelikli Bütünleştirme

En üst düzeyden başlanır. Birim şemasında bulunan her dal soldan sağa olmak üzere ele alınır. Bir dala ilişkin bütünleştirme bitirildiğinde diğer dalın bütünleştirilmesi başlar.

AŞAĞIDAN YUKARIYA SINAMA ve BÜTÜNLEŞTİRME

Önce en al düzeydeki işçi birimleri sınanır ve bir üstteki birimle sınanması gerektiğinde bu üst bileşen, bir ‘sürücü’ ile temsil edilir, koçan ile aynı özelliklere sahiptir.

SINAMA PLANLAMASI

Sınama yönteminde yer alması gereken başlıklar: Sınama konusu, sınama etkinlikleri ve zamanlama, temel sınama teknikleri, destek etkinlikler, kaynaklar ve sorumluluklar, personel ve eğitim gereksinimleri, sınama yaklaşımı, riskler ve çözümler, onaylar.

Sınama ayrıntılarında yer alması gereken başlıklar: sınanacak sistemler, girdiler ve çıktılar, sınamaya başlanma koşulları, girdilerin hazır olması, ortam koşulları, kaynak koşulları, sınama tamamlama kıstası, sınama geçme-kalma kıstası, sınama askıya alınma kıstası ve sürdürme gerekleri, sınama sonuçları.

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

YAŞAM DÖNGÜSÜ BOYUNCA SINAMA ETKİNLİKLERİ

Planlama aşamasında genel sınama planı oluşturulur. Söz konusu plan tüm sınama etkinliklerini çok genel hatlarıyla tanımlar ve sınama planlamasında verilen bilgileri içerir.

Çözümleme aşamasında, sistemler ve alt sistemler ortaya çıkarılır ve sınama planı alt sistemler bazında ayrıntılandırılır.

Tasarım aşamasında tüm yazılım modüllerinin ortaya çıkarıldığı aşamadır. Bu aşamanın başlangıcında yazılım modülleri için sınama planı detaylandırılır ve sınama belirtimleri hazırlanır. Söz konusu belirtimler, kullanıcı kılavuzları ile birlikte eğitim için temel bilgileri oluşturur.

Gerçekleştirim aşamasında teknik sınamalar yapılır, sınama raporları hazırlanır ve kullanıcı sınayıcıları eğitilir.

Bakım aşamasından hemen sonra kullanıcı sınamaları yapılarak sınama raporları hazırlanır.

Sınama sırasında bulunan her hata için değişiklik kontrol sistemine yazılım değişiklik isteği türünde bir kayıt girilir.

Onulmaz Hatalar: BT projesinin gidişini bir ya da birden gazla aşama gerileten ya da düzeltilmesi mümkün olmayan hatalardır.

Büyük Hatalar: Projenin kritik yolunu etkileyen ve önemli düzeltme gerektiren hatalardır. Küçük Hatalar: Projeyi engellemeyen ve giderilmesi az çaba gerektiren hatalardır.

Şekilsel Hatalar: Heceleme hatası gibi önemsiz hatalardır.

PL/SQL : Oracle tarafından geliştirilen veritabanı sistemleri özel bir dildir. Yapısal dillere ait özelliklerin SQL’e eklenmesiyle ortaya çıkmış bir dildir. SQL’den farkı ise **SQL**, verileri seçmek ve işlemek için veri odaklı bir dil olmakla birlikte, **PL SQL**, uygulamalar oluşturmak için kullanılan bir yordam dalıdır. **SQL**, bir defada bir deyimi yürütürken, **PL**' de **SQL** kod bloğu yürütülebilir. **SQL**, **PL SQL** 'in usule uygun olarak bildirimsel niteliktedir.

1. HAFTA

Sınama işlemleri bitirilen yazılımın kullanıcı alanında yüklenmesi ve uygulamanın başlatılması gerekmektedir.

yazılım kullanıma geçtikten sonra yaşam döngüsünün en önemli ve hiç bitmeyecek aşaması olan bakım aşaması başlar.

izleye kesimlerde kurulum ve bakım aşamasında yapılması gerekenler açıklanmaktadır. bakım bölümüne ilişkin yapılan açıklamalarda IEEE 1219-1998 standardı dikkate alınmıştır.

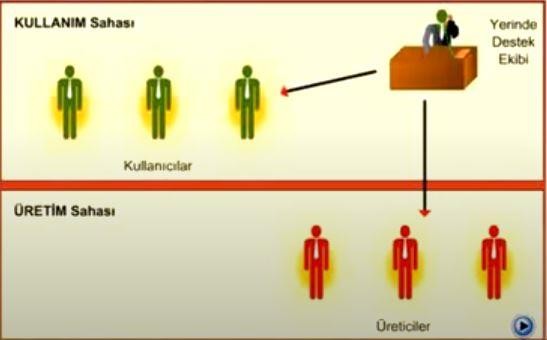
Kurulum

sınanmış yazılımların kullanıcı sahasına aktarılması ve yazılımın gerçek yaşamda uygulanmasının başlatılması için yapılan işlemler kurulum işlemleri olarak tanımlanır.

* + donanım kurulumu
  + sistem yazılımları kurulumu
  + veritabanı kurulumu
  + uygulama yazılımları kurulumu
  + eğitim
  + yerinde destek

Yerinde Destek Organizasyonu

* + yerinde destek ekibi kullanıcı alanında yerleşik olarak bulunan gerekli sayıda elemandan oluşan bir ekiptir. temel görevleri;
  + kullanıcıları ziyaret ederek sorunlarını belirlemeye çalışmak, giderilebilen kullanıcı sorunlarını gidermek ve giderilemeyenleri üretim sahasında uygulama yazılım destek ekibine iletmek
  + Kullanıcıya iş başında uygulama eğitimi vermek
  + kullanıcı sınama günlükleri toplamak
  + Yapılan tüm işlemleri konfigürasyon veritabanına kaydetmek biçimindedir



Bakım Süreç Modelleri;

* + sorun tanımlama süreci
  + çözümleme süreci
  + tasarım süreci
  + gerçekleştirim süreci
  + Sistem sınama süreci
  + kabul sınaması süreci
  + kurulum süreci başlıkları altında incelenir.

Sorun tanımlama süreci



Bakım tanımı: bakım işletimi alınan yazılımın sağlıklı olarak çalışması ve ayakta kalabilmesi için yapılması gereken çalışmaların bütünü olarak tanımlanır. Uygulamada çalışan bir yazılımın 3 tür bakım gereksinimi bulunur.

* + Düzeltici bakım
  + uyarlayıcı bakım
  + en iyileyici bakım

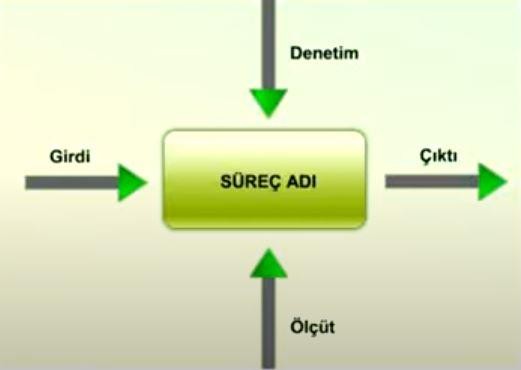
Düzeltici bakım; yazılım sınanması teorik olarak tamamen yapılabilirse de pratikte pek sağlanamaz bu nedenle yazılımda her an bir hata ile karşılaşabiliriz hataların düzeltilmesi için de düzeltici bakım devreye girer.

Uyarlayıcı bakım; uygulama yazılımları işletme ya da kuruluşların günlük yaşamda yaptıkları işleri bilgisayar ortamında yapmalarını sağlayan araçlardır. işletme ya kuruluşlarda yaşanan değişikliklerin o kuruluşun işlerini bilgisayar yardımı ile yapmalarını sağlayan uygulama yazılımlarına da yansıtılması gerekir. bu yansıtma işlemi uyarlayıcı bakım olarak tanımlanır.

En iyileyici bakım; zaman zaman uygulama yazılımlarının çalışma performanslarının iyileştirilmesi amacıyla yapılan çalışmalar en iyileyici bakım olarak tanımlanır.

Bakım süreç modeli:

IEEE 1219 standardı tarafından önerilen bakım süreç modeli şekilde belirtilen şablon kullanarak bakım süreçlerini tanımlamaktadır.



Bakım süreç modelinin süreçleri:

* + Sorun tanımlama/sınıflandırma
  + Çözümleme
  + Tasarım
  + Gerçekleştirim
  + Sistem sınama
  + Kabul sınama
  + Kurulum

Bakım süreci yazılım yaşam döngüsü çekirdek adımlarının bir anlamda yinelenmesinden oluşmaktadır.

Sorun tanımlama süreci;



Girdi; temel girdi bakım isteğidir. bakım istekleri:

* + sistemde beklenen ve yeni düzenlemelere ilişkin değişiklikler
  + Yeni işlev istekleri
  + Yazılımda bulunan yanlışların düzeltilme istekleri
  + performans artırımına ilişkin istekler
  + yeni iş yapma türlerine ilişkin istekler
  + Teknolojinin zorlaması sonucu oluşan istekler örnek verilebilir

İşlem/Süreç:

Yazılım bakım isteği oluştuğunda yapılması gerekenler;

* + Değişiklik isteğini bir tanım numarası verilmeli
  + değişiklik türünün sınıflandırılması
  + değişik isteğinin kabul red ya da daha ayrıntılı incelenmesi yönünde karar alınması
  + değişiklik ile ilgili zaman boyut işgücü kestirme yapılması
  + değişiklik isteğinin diğerleri ile birlikte zaman ve iş planına kaydedilmesi

Birçoğunda değişiklik isteyen kişi, kullanıcı temsilcileri, yazılım mühendisleri, konu uzmanlarıdır.

Denetim: sorun tanımlama aşamasında değişiklik isteğinin daha önceden yapılıp yapılmadığını denetlemeli ve tek olduğu belirlenmelidir. bu amaçla daha önceki değişiklik istekleri taranır.

Çıktı:

Temel çıktısı doğrulanmış, geçerlenmiş ve karar verilmiş bakım isteğidir. Veritabanında saklanan isteğin ayrıntıları;

* + Sorun ya da yeni gereksinim tanımı
  + sorun ya da yeni gereksinim değerlendirilmesi
  + Başlangıç önceliği
  + geçerleme verisi
  + Başlangıç kaynak gereksinimi
  + mevcut ve gelecekte kullanıcılar üzerinde etkileri
  + yararlı ve aksak yönleri

Ölçüt: oğlum tanımlama sırasında kullanılabilecek ölçütler;

* + Bakım isteklerinde kabul edilmeyen madde sayısı
  + gelen bakım istekleri sayısı
  + sorun geçerleme için harcanan kaynak ve zaman biçimindedir.

Çözümleme süreci; veritabanında saklanmış ve geçerlenmiş bakım isteğini girdi olarak alır, projeye ilişkin bilgi ve belgeleri kullanarak söz konusu isteğinin yerine getirilmesi için gerekli genel planı yapar.

Resim 5

Girdi;

* + geçerlenmiş bakım isteği
  + başlangıç kaynak gereksinimleri ve diğer veriler
  + mevcut proje ya da sistem bilgi ve verileri biçimindedir.

İşlem/Süreç: olurluk aşaması ve ayrıntılı çözümleme aşamasıdır.

Bu süreç, değişikliğin etkisi,

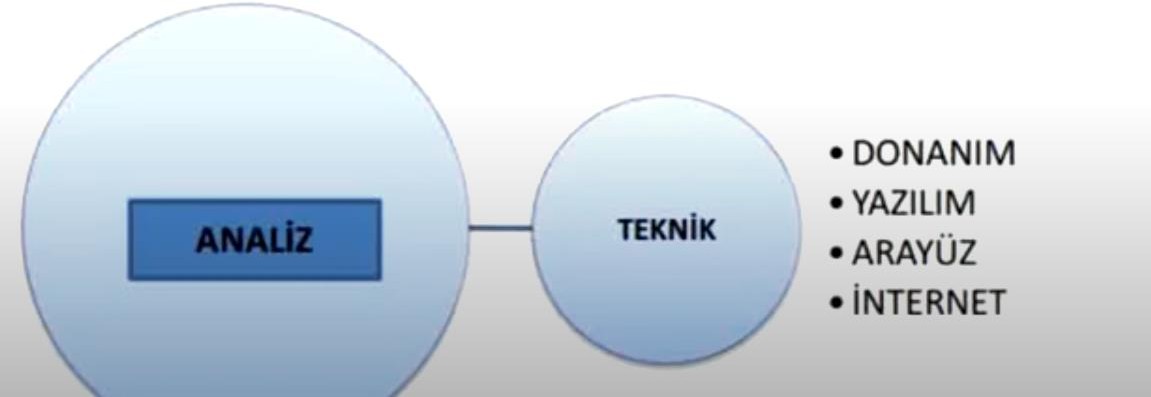
* + Prototipleme içeren seçenek çözümler
  + dönüştürme gereksinimlerinin çözümleri
  + güvenlik ve emniyet zorunlulukları
  + İnsan faktörleri
  + kısa ve uzun verimli faaliyetler
  + değişikliği yapmanın yararları içerir.

# GEREKSİNİM VERİ TOPLAMA

**-**Yazılım mühendisliği projelerinin geliştirilmesi sürecinde en önemli unsurlardan biriside mevcut sistemin incelenmesidir. Genel olarak müşteri ile görüşmeler ve mevcut sistemin incelenmesi sonucunda gereksinim analizi dökümanı üretilir

*Mevcut sistemin incelenmesi sırasında kullanılabilecek temel yöntemler aşağıdaki gibi özetlenebilir;*

## Sorma

1. **Psikolojik Türetme Teknikleri 3-İstatiksel Teknikler**

**NOT:**Yaptığımız donanım donanım ürünlerini karşılıyor mu ihtiyaç var mı ? İşletim sistemleri? Lisanslar mevcut mu ? Vüris programları ? Daha önce kullanılmış uygulama var mı (programla combined uygulama var mı )? Mobil mi Masaüstümü ?Ekran tasarımı nasıl olsun

? Sistemin yükünü karşılayacak ağ yapısı var mı ? Güncelleme karşı internet üzerinden müdahale edilebilir mi ? vs gibi sorular kullanıcıya sorular

# KOD BANKASI PROJESİ VERİ SÖZLÜĞÜ ÖRNEĞİ

**-**Kod Bankası Projesi en önde gelen hedefi kod aramalarında en çok ziyaret edilen ; kodlara hızlı erişim sağlayan ve güvenilir içeriği ile yazılım sektöründeki kullanıcıların

vazgeçemeyeceği bir web kaynak sayfasını oluşturmaktadır.(Programcıya örnek ve yol gösterici olabilir)

## İşlev Gereksinimleri

**-**Müşteri kullanıcı adı ve şifre , isim ,e-mail adresi ile kullanıcıların oturuma girmesini ister

-Kullanıcıların , farklı programlama dillerinde kod yükleyebilmesini ister

-Kullanıcıların yükledikleri kodları çalıştırabilmelerini ister

-Kullanıcıların veri tabanına kolay ve hızlı bir şekilde tarama yapabilmelerini ister KOD BANKASI PROJESİ VERİ SÖZLÜĞÜ örneği (sitede paylaşılanlar)



## Not: Burdaki asıl amaç github gibi siteleri anlatıyor demek isteği programcıya destek siteleri

**! HER PROJE FİKİRLE BAŞLAR**

