Yapay Sinir Ağları

Vizeden sonraki konular:
2019 (canan Hoca)
Ment İNCİDELEN

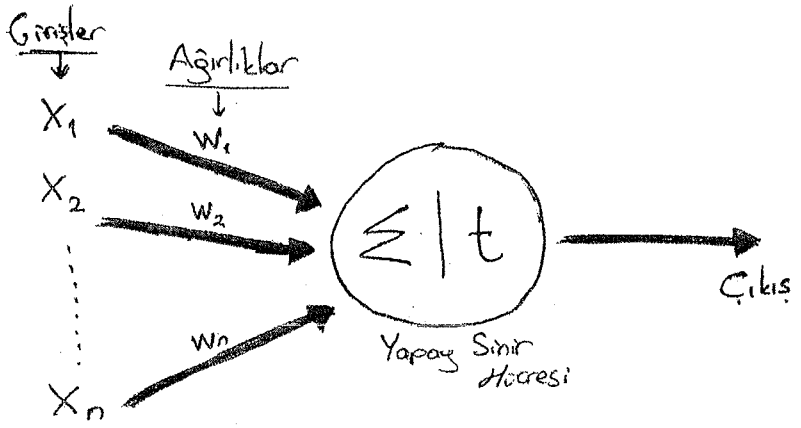
Belirli bir sürede bir hücreye gelen girişlerin değeri, belirli bir eşik değerine ulaştığında hücre tepki üretir. Hücrenin tepkisini arttırıcı yönde girişler uyarıcı, azaltıcı yönde girişler ise önleyici girişlerdir.

Yapay Sinir Ağlarının Genel Özellikleri:

- Doğrusal olmama özelliği (non-lineer)
- Öğrenme özelliği
- Genelleme özelliği
- Uyarlanabilirlik özelliği
- Hata toleransı
- Donanım ve hız

★ Final Sorusu (2019)

Yapay Sinir Hücresi:

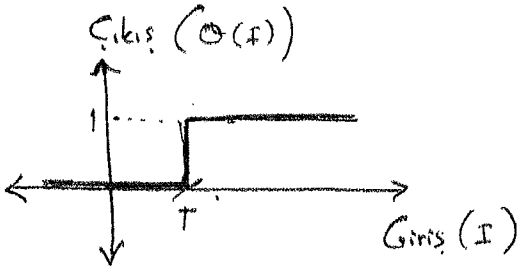


$$\Sigma = X_1 \cdot W_1 + X_2 \cdot W_2 + \ldots\ldots\ldots + X_n \cdot W_n$$

t = Eşik Fonksiyonu

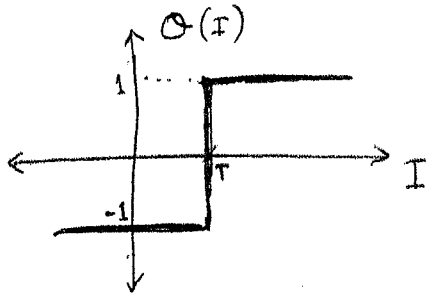
Eşik Fonksiyonları

1) Birim Basamak Fonksiyonu



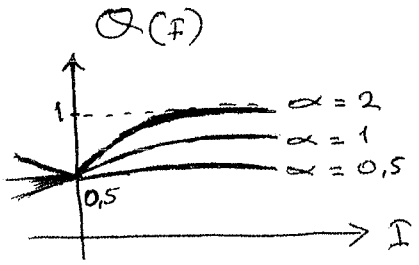
$$O(I) = \begin{cases} 1, & I > T \\ 0, & I \leq T \end{cases}$$

2) Signum Fonksiyonu



$$O(I) = \begin{cases} +1, & I > T \\ -1, & I \leq T \end{cases}$$

3) Logistic Fonksiyonu



$$O(I) = \frac{1}{1+e^{-\alpha \cdot I}}$$

4) Sigmoid Fonksiyonu

★ Çözümlerde kullanılacak fonk. (2019 Cfinal)

$$O(I) = \frac{1}{1+e^{-I}}$$

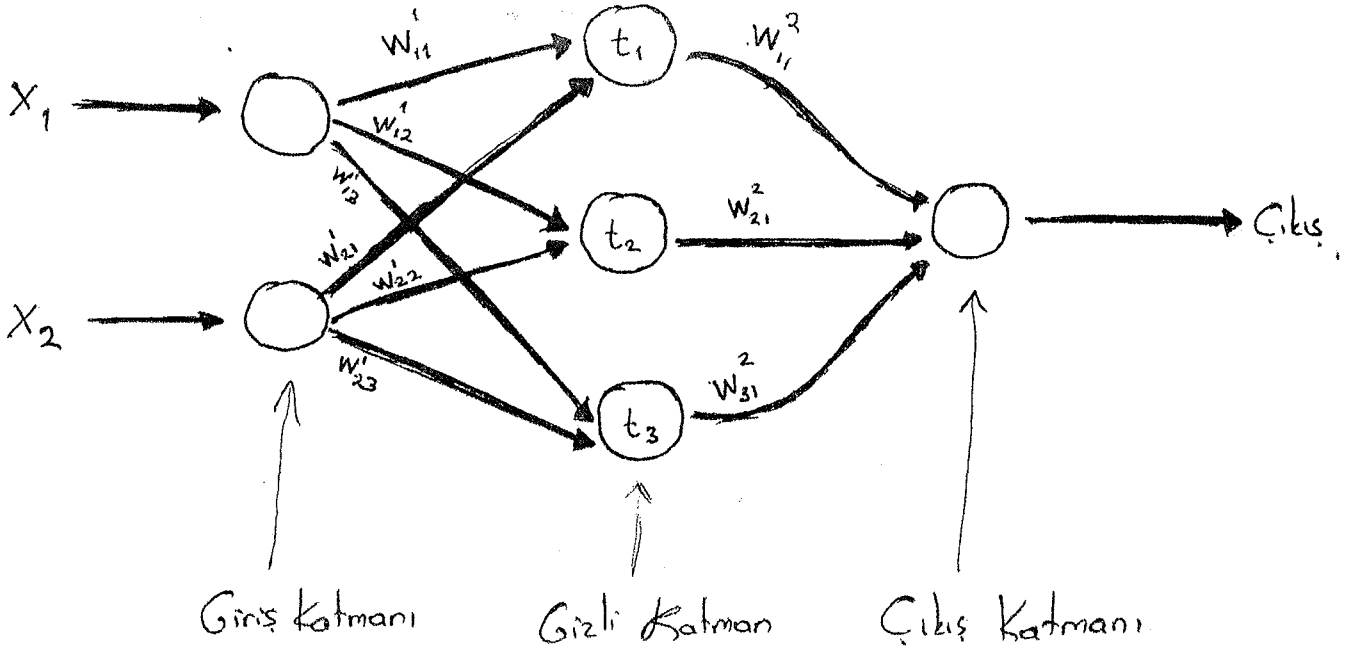
5) Doğrusal Fonksiyon

★ Çözümlerde kullanılacak fonk. (2019)

$$O(I) = I$$

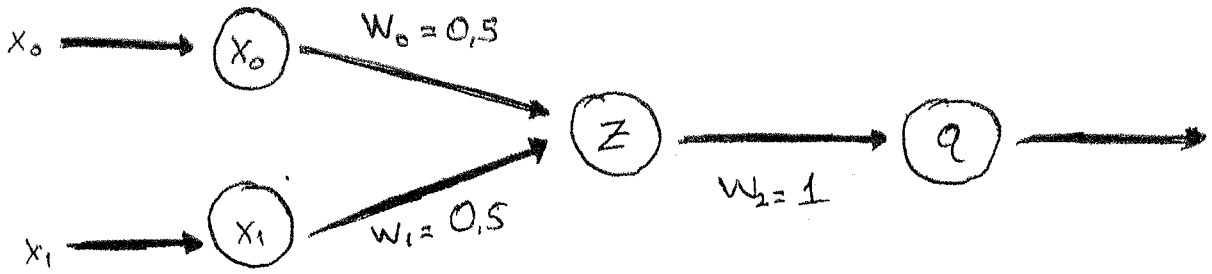
İleri Yönlü Yapay Sinir Ağları

İleri yönlü yapay sinir ağlarında hücreden çıkan bağlar hep bir sonraki hücreye girer



OR (Veya) Probleminin Yapay Sinir Ağları ile Çözümü

★ Final Sorusu (2019)



Aktivasyon fonksiyonu (Z) = if $z > 0$ then 1, else 0

$X_0 \cdot W_0 + X_1 \cdot W_1$

$X_0 = 0$, $X_1 = 0$ için $Z = 0 \cdot 0,5 + 0 \cdot 0,5 = 0 \xrightarrow{\text{Akt. Fonk}} Q = 0$

$X_0 = 0$, $X_1 = 1$ için $Z = 0 \cdot 0,5 + 1 \cdot 0,5 = 0,5 \xrightarrow[0,5 > 0]{\text{Akt. Fonk.}} Q = 1$

$X_0 = 1$, $X_1 = 0$ için $Z = 1 \cdot 0,5 + 0 \cdot 0,5 = 0,5 \xrightarrow[0,5 > 0]{\text{Akt. Fonk}} Q = 1$

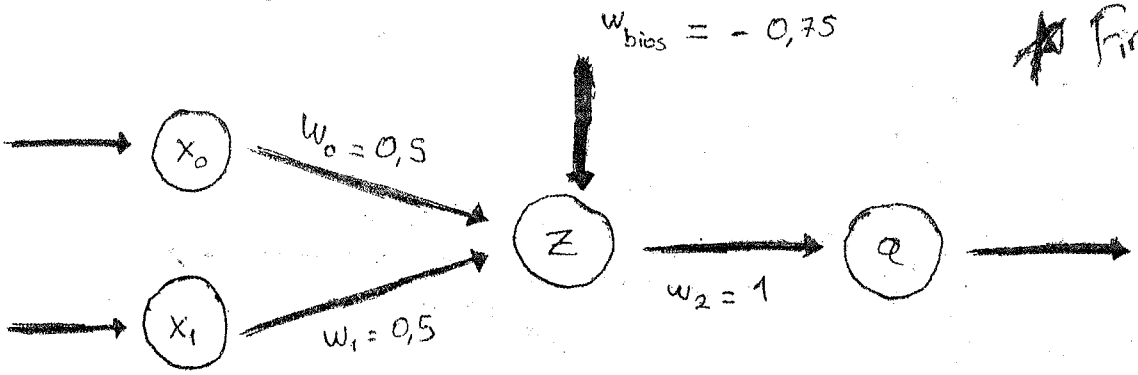
$X_0 = 1$, $X_1 = 1$ için $Z = 1 \cdot 0,5 + 1 \cdot 0,5 = 1 \xrightarrow[1 > 0]{\text{Akt. Fonk}} Q = 1$

buna göre:

X_0	X_1	Q
0	0	0
0	1	1
1	0	1
1	1	1

AND (Ve) Probleminin Yapay Sinir Ağları İle Çözümü

★ Final Sorusu (2019



> Aktivasyon Fonksiyonu (z) = if $z > 0$ then 1, else 0

$x_0 . w_0 + x_1 . w_1 + w_{bios}$

$x_0 = 0$, $x_1 = 0$ için $z = 0.0,5 + 0.0,5 + (-0,75) = -0,75 \xrightarrow{\text{Akt. Fonk.}} Q = 0$

$x_0 = 0$, $x_1 = 1$ için $z = 0.0,5 + 1.0,5 + (-0,75) = -0,25 \xrightarrow{\text{Akt. Fonk.}} Q = 0$

$x_0 = 1$, $x_1 = 0$ için $z = 1.0,5 + 0.0,5 + (-0,75) = -0,25 \xrightarrow{\text{Akt. Fonk.}} Q = 0$

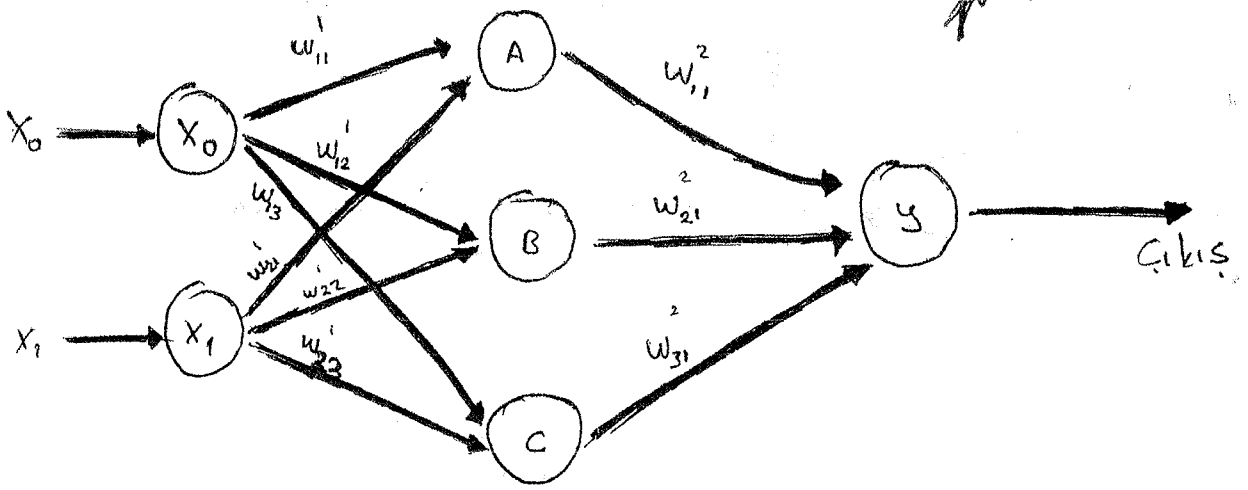
$x_0 = 1$, $x_1 = 1$ için $z = 1.0,5 + 1.0,5 + (-0,75) = 0,25 \xrightarrow{\text{Akt. Fonk.}} Q = 1$

buna göre

x_0	x_1	Q
0	0	0
0	1	0
1	0	0
1	1	1

İleri Yönlü YSA Örnek:

★ Finalde Soruldu (2019)



Giriş Değerleri : $X_0 = 0,2$ $X_1 = 0,3$

Ağırlıklar : $w'_{11} = 0,4$ $w'_{12} = 0,1$ $w'_{13} = 0,6$

$w'_{21} = 0,2$ $w'_{22} = 0,5$ $w'_{23} = 0,7$

$w^2_{11} = 0,9$ $w^2_{21} = 0,8$ $w^2_{31} = 0,1$

→ Ara katman için Sigroid, çıkış için doğrusal fonksiyonu kullanırsak:

✱ 1. ara düğüm için : $X_0 \cdot w'_{11} + X_1 \cdot w'_{21}$

$= 0,2 \cdot 0,4 + 0,3 \cdot 0,2 = 0,4$

$A = \frac{1}{1+e^{-x}}$ uygulanırsak ⟶ $A = \frac{1}{1+e^{-0,4}} = 0,53$

✱ 2. ara düğüm için : $X_0 \cdot w'_{12} + X_1 \cdot w'_{22}$

$= 0,2 \cdot 0,1 + 0,3 \cdot 0,5 = 0,17$

$B = \frac{1}{1+e^{-0,17}} = 0,54$

✱ 3. ara düğüm için : $x_0 \cdot w'_{13} + x_1 \cdot w'_{23}$

$= 0,2 \cdot 0,6 + 0,3 \cdot 0,7 = 0,33$

$C = \frac{1}{1+e^{-0,33}} = 0,58$

Buna göre . $A = 0,53$ $B = 0,54$ $C = 0,58$

✱ Çıkış düğümü için : $A \cdot w^2_{11} + B \cdot w^2_{21} + C \cdot w^2_{31}$

$= 0,53 \cdot 0,9 + 0,54 \cdot 0,8 + 0,58 \cdot 0,1 = 0,967$

$y =$ (doğrusal fonksiyon uygulanacağından) $0,967$

⇒ YAZILIM MÜHENDİSLİĞİ ⇐

* YAZILIM MÜHENDİSLİĞİ GERÇEKLİĞİ
* YAZILIM GELİŞTİRME YAŞAM DÖNGÜSÜ
* PLANLAMA
* SİSTEM ANALİZİ
* TASARIM
* GERÇEKLEŞTİRİM
* YAZILIM DOĞRULAMA VE GEÇERLEME

- HELİN YARDIMCI -

BÖLÜM-1 / YAZILIM MÜHENDİSLİĞİ BERGEĞİ

Yazılım Geliştirme İstatistikleri

- Tipik bir yazılım projesinin geliştirilmesi 1-2 yıl, en az 500.000 satır kod
- Projelerin %70'i 80'i başarıyla tamamlanıyor.
- Bu süreçte her birey günde 10 satırdan az kod yazıyor
- Geliştirme sürecinde her 1000 kaynak kod satırında 50-60 hata bulunuyor.

Yazılım Problemleri (Krizleri)

- Tasarlanan zamanın gerisinde kalma
- Bütçeyi aşma
- Düşük kalite
  - * Güvenilir olmayan yazılım
  - * Yetersizlik
  - * Sürdürülebilirliğin sağlanmasındaki zorluk

Yazılım

= Mantık (algoritma) + Veri (test verisi) + Belge (dokümanlar) + İnsan (kullanıcı) + Program (kod)

* Yazılımın temel amacı "veri"yi "bilgi"ye dönüştürmektir.
* Yazılım üretimi bir mühendislik disiplini gerektirir

Yazılım vs Donanım

* Yazılım geliştirilir ve donanım üretilir.
* Yazılım eskimez.
* Oysa, her donanımın belli bir ömrü vardır (yeniyle değiştirilir)
* [illegible]
  - Personel kullanımı → Donanım – Doğru orantılı
    Yazılım – Ters orantılı

Tipik Bir Yazılım Üretim Ortamı

- Değişik yeteneklerde bir çok personel
- Yazılım çıktısı ile ilgilenen kullanıcılar
- Yenilikçe tepki gösteren kullanıcılar ve yöneticiler
- Yeterince tanımlanmamış kullanıcı beklentileri
- [illegible]
- Mevcut yazılımlardaki kalitesizlik
- Yüksek üretim maliyeti

Yazılım Mühendisliği

- Yazılım mühendisliği bir yöntemler, teknikler ve araçlar kümesi olarak değerlendirilebilir.
- Yazılım Mühendisliğinin Hedefi : Yazılım üretimindeki karmaşıklıkları gidermek
- Yazılım üretimi tek kişiyle değil takım halinde yapılmalıdır.
- Hedef : En az maliyet, yüksek kalite

Yazılımda Kalite

Güvenilirlik, Değiştirilebilirlik, Taşınabilirlik, Bakımlanır, Esneklik, Geçerlik, Anlaşılabilirlik, Kullanılabilirlik, Sınanabilirlik

BÖLÜM-2 / YAZILIM GELİŞTİRME YAŞAM DÖNGÜSÜ

- Yazılımın hem üretim, hem de kullanım süreci boyunca geçirdiği tüm aşamalar yazılım geliştirme yaşam döngüsü olarak tanımlanır.
- Döngü içerisinde herhangi bir aşamada geriye dönmek veya tekrar ilerlemek söz konusudur.
- ~~Tek yönlü ve doğrusal değildir~~

Yazılım Yaşam Döngüsü Temel Adımları

1) Planlama => Proje planının oluşturulduğu aşamadır.
2) Analiz => Sistem gereksinimleri ve temel sorunlar [illegible]
3) Tasarım => Yazılım sisteminin temel yapısının oluşturulduğu aşamadır.
4) Gerçekleştirim => Kodlama, test etme, kurulum
5) Bakım => Hata giderme ve yeni eklentiler yapma (teslimden sonra)

Yazılım Süreci Modelleri

1) Gelişigüzel Model
2) Barok Modeli
3) Çağlayan Modeli
4) V Modeli
5) Helezonik (Spiral) Model
6) Evrimsel Model
7) Artırımsal Model
8) Araştırma Tabanlı Model

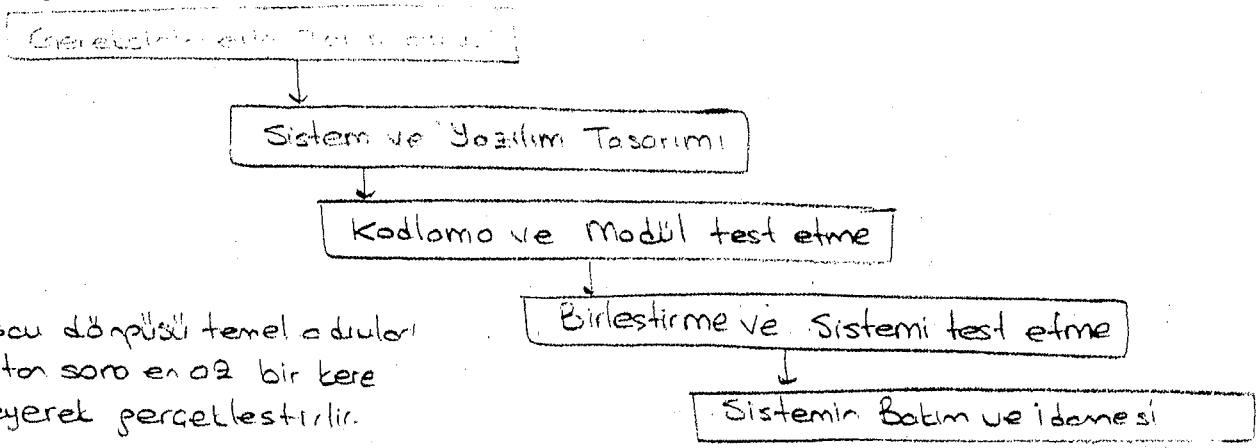
Gelişigüzel Model

- Herhangi bir model ya da yöntem yok.
- Geliştiren kişiye bağlı
- İzlenebilirliği ve bakımı oldukça zor
- Genellikle tek kişilik üretim ortamı
- Basit Programlama

Barok Modeli

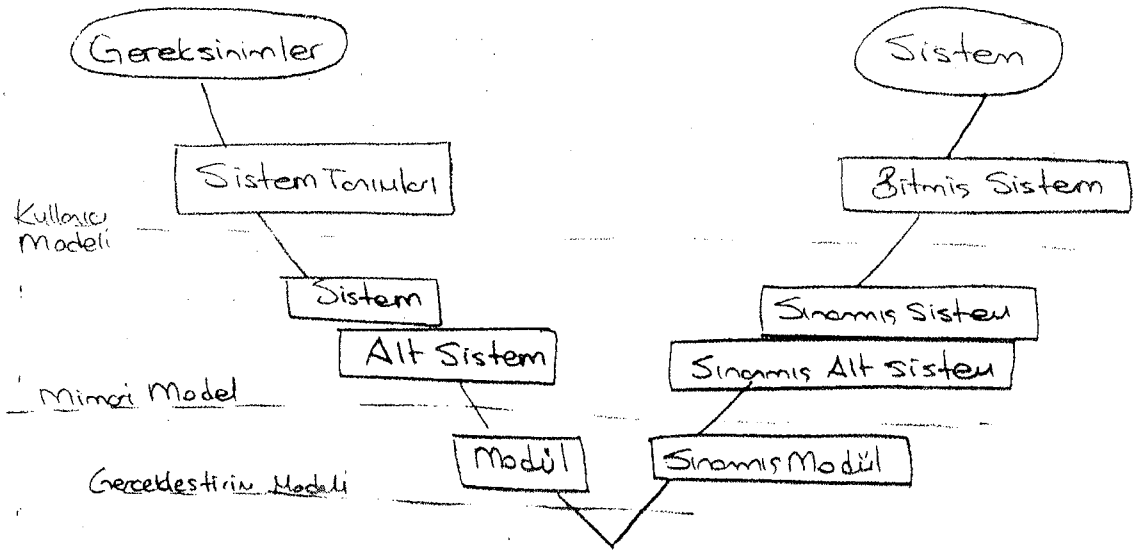
- ~~Yasam döngüsü temel adımlarını doğrusal olarak geliştirir.~~
- Aşamalar arası geri dönüşlerin nasıl yapılacağı tanımlı değil.
- Gerçekleştirim aşamasına daha fazla ağırlık veren bir model olup günümüzde kullanımı önerilmemektedir.

Çağlayan (Şelale) Modeli



- Yasam döngüsü temel adımları baştan sona en az bir kere işleyerek gerçekleştirilir.
- Üretimi az zaman gerektiren yazılım projeleri için uygun bir modeldir.
- Bu modelin kullanımı günümüzde giderek azalmaktadır.
- Barok modele göre geri dönüşler iyi tanımlanmıştır.
- Yazılımda belirsizlik yoksa ve yazılım üretimi çok zaman almayacaksa uygundur.

V Süreç Modeli



- Sol taraf üretim, sağ taraf sınama işlemidir.

Kullanıcı Modeli

Geliştirme sürecinin kullanıcı ile olan ilişkileri tanımlanmakta.

Mimari Model

Sistem Tasarımı

Gerçekleştirim Modeli

Yazılım modüllerinin kodlanması

* Belirsizliklerin az, iş tanımlarının belirgin olduğu projeler için uygundur.
* Model, kullanıcının projeye katkısını arttırmaktadır.

Helezonik (Spiral) Modeli

1) Planlama
2) Risk Analizi
3) Üretim (Ara ürün)
4) Kullanıcı Değerlendirmesi (Ara ürün değerlendirmesi)

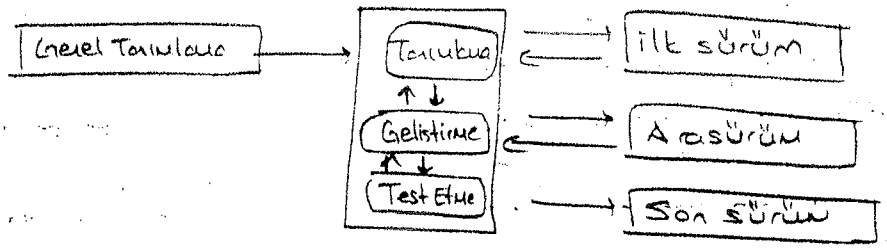
Avantajları

1) Kullanıcı katkısı
2) Yönetici Bakış
3) Yazılım Geliştirici Bakışı

- Risk analizi ön plana çıkmıştır.
- Her döngü bir ürün ifade eder.
- Prototip bir yaklaşım verir.

Evrimsel Geliştirme Süreç Modeli

- İlk tam dlaekli modeldir.
- Coğrafik olarak geniş alana yayılmış organizasyonlar için önerilmektedir.
- Her aşamada üretilen ürünler, tam işlevseldir.



Artırımsal Geliştirme Süreç Modeli

- Üretilen her yazılım sürümü birbirini kapsayacak
- Uzun zaman alabilecek ve sistemin eksik işlevlikle çalışabileceği türdeki projeler bu modele uygun olabilir.
- Bir tarafta kullanım, diğer taraftan üretim yapılır.

Araştırma Tabanlı Süreç Modeli

- Araştırma ortamları bütünüyle belirsizlik üzerine çalışan ortamlardır.
- Yapılan işlerden edinilecek sonuçlar belirgin değildir.
- Geliştirilen yazılımlar genellikle sınırlı sayıda kullanılır ve kullanım bittikten sonra ise yararsız hale gelir ve atılır.
- Sabit fiyat sözleşmelerinde uygun değildir. (Belirsizlikten dolayı)

BÖLÜM-3 / PLANLAMA

Planlama

- Yazılım geliştirme sürecinin ilk aşaması
- Başarılı bir proje geliştirmek için projenin resmi çıkarılır

Planlama Aşamalarındaki İşlemler

* Proje Kaynaklarının belirlenmesi
* Proje Maliyetlerinin Kestirilmesi
* Proje Ekip Yapısının Oluşturulması
* Ayrıntılı Proje Planının yapılması
* Projenin izlenmesi

- Proje planı; sürekli olarak kullanılır ve güncelleştirilebilir, gözden geçirilebilir bir belgedir.

Proje Kaynakları

- İnsan kaynakları
- Donanım kaynakları
- yazılım kaynakları

Planlama: bu kaynakların tanımını yapar ve zaman kullanımı, görev süreleri, edinilme zamanlarını planlar.

- Hangi tür elemanların, hangi süre ile projenin hangi aşamasında yer alacağını belirler.

Donanım Kaynakları

- Ara Bilgisayarlar
- Sunucular (Web, E-posta, Veri Tabanı)
- Kullanıcı Bilgisayarları (PC)
- Yerel Alan Ağı (LAN) Alt Yapısı
- Geniş Alan Ağı (WAN) Alt yapısı

* Yazılımın geliştirileceği ortam, gerçek kullanım ortamı dışında olmalıdır. Öte yandan, geliştirme ve uygulama ortamları aynı konfigürasyonda olursa, ileride kurulum sırasındaki taşıma sorunlarını büyük ölçüde giderecektir.

Yazılım Kaynakları

- Bilgisayar Destekli Tasarım (CAD)
- Bilgisayar Destekli Mühendislik (CASE)

Yazılım Kaynakları

- İş Sistemleri Planlama araçları
- Proje Yönetim araçları
- Analiz ve tasarım ağaçları
- Programlama araçları
- Test araçları
- Prototipleme ve simülasyon araçları
- Bakım araçları
- Destek araçları

Proje Maliyetleri

Maliyet Kestirimi; Bir yazılım için gerekebilecek iş gücü ve zaman maliyetlerinin üretimden önce belirlenebilmesi için yapılan işlemlerdir.

Kullanılan Unsurlar

- Geçmiş projelere ilişkin bilgiler
- Proje ekibinin deneyimleri
- İzlenen geliştirme modeli

Maliyet Yönetimi sayesinde;

- Gecikmeler önlenir.
- Geliştirme süreci kolaylaştırılır.
- Etkin kaynak kullanımı
- İş zaman planı etkin
- Ürün sağlıklı olarak fiyatlandırılır.
- Ürün zamanında ve belirlenen bütçeyle bitirilir.

Maliyet Kestirim Yöntemleri

1) Projenin boyutuna göre
2) Projenin büyüklüğüne göre
3) Uygulama biçimlerine göre
4) Değişik ortamlarda kullanılabilirlik
5) Yöntemlerin yapılarına göre

Proje Sınıfları

Açık Projeler: Boyutları küçük, Deneyimli personel tarafından yönetilmiş.

Yarı Gömülü: Hem bilgi boyutu hem donanım sürme boyutu olan projeler

Gömülü Projeler: Donanım sürmeyi hedefleyen projeler

Proje Ekip Yapısı Oluşturma

PANDA proje Ekip yapısı temel olarak her proje biriminin doğrudan proje yönetimine bağlı olarak çalışması ve işlevsel bölümleme esasına göre düzenlenir.

Temel Bileşenler:

- Proje Denetim Birimi
- Kalite Yönetim Birimi
- Teknik Destek Birimi
- Proje Yönetim Birimi
- Proje Ofisi
- Yazılım Üretim Birimi
- Eğitim Birimi
- Uygulama Destek Birimi

BÖLÜM-4 / SİSTEM ANALİZİ

- Sistem Analiz çalışması, üretim sürecinin başlayacıdır.
- Amaç: Mevcut Sistemin nasıl çalıştığının araştırılması

Gereksinim

Gereksinim sistemin ya da işlevlerinin nasıl yerine getirileceği ile ilgili değildir. Ne olduğu ile ilgilidir.

- hangi veritabanı
- hangi tablolar
- ne kadar bellek kullanılıyor

} bunlar gerçekleştirim aşamasında ele alınır.

İşlevsel Gereksinim

Sistemin herhangi bir durum karşısında davranışını, nasıl iletişim kuracağını belirler

Gereksinim Türleri

- Fiziksel Gereç
- Arayüzler
- Kullanıcı ve insan etmeni
- İşlevsellik
- Belgeleme
- Veri
- Kaynaklar
- Güvenlik
- Kalite Güvencesi

Gereksinim Özellikleri

Gereksinimler üçüncü kişilere hizmet eder.

- Geliştiricilerin, müşterilerin sistemin nasıl çalışmasını istediklerini anlamalarını sağlar.
- Gereksinimler, sonuç sistemin ne özellikte ve işlevsellikte olacağını söylerler.
- Gereksinimler sınama ekibine, kullanıcıyı, sunulan sistemin istenen sistem olduğuna ikna etmek için neler göstermeleri gerektiğini söylerler.

Sistem Çözümleme Aşaması (Çözümü)

- Geliştirilecek yazılımla ilgili olarak:
  * Tüm gereksinimlerin araştırılması
  * Tanımlanması
  * Ortaya çıkarılması ve açıklanmasıdır

Süreç / İşlem Modelleme Yöntemleri

- Veri Akış Diyagramları (DFD)
- Süreç Tanımlama Dili (PDL)

Veri - Akış Diyagramı

- Yukarıdan-aşağıya bir yaklaşımla oluşturulur.
- Sistem önce en genel biçimiyle ele alınır, yalnızca dışsal ilişkileri incelenir.
- Daha sonra sistemin iç yapısındaki süreçler modellenir.

Kapsam Diyagramı: Dışsal ilişkilerini gösterir.

Genel Bakış Diyagramı: Ana işlevleri ve bu işlevlerle ilgili veri kaynaklarını ve veri depolarını içerir.

Detay Diyagramı: Ayrıntı düzeyinde detaylandırılır.

Veri Akış Diyagramı Neyi Gösterir?

- Sistemin durağan yapısını,
- Sistemin süreçlerini ve süreçler arasındaki veri akış ilişkisini,
- Bilgi sisteni için gerekli olan ana veri depolarının neler olduğunu ve hangi süreçler tarafından kullanıldığını
- Bilgi sisteminin süreçlerini yukarıdan aşağıya ayrıştırma ile gösterir.

Veri Akış Diyagramı Neyi Göstermez?

- Bilgi sisteminin süreçlerinin zamana ilişkin durumunu ve bu duruma ilişkin bilgileri göstermez.
- Bilgi sistemi süreçlerinin kendi aralarındaki ilişkisini göstermez.
- Veri kaynakları ve depoları için ayrıntı içermez.

Kullanıcı Arayüz Prototipleme (KAP)

- Ekran tasarımı için kullanıcıdan onay alınması esastır.
- Gereksinimlerin kesinleştirilmesini kolaylaştırır.

Özellikleri

- Ayrılan zaman sistem analizi için ayrılan zamanın %5'ini aşmamalıdır.
- Her bir özellik bir kez prototiplenebilir.
- Hiç bir işlem işlemeli(r).

BÖLÜM-5 / TASARIM

Tasarım Kavramları

* SOYUTLAMA (abstraction)

Detayları gizleyerek yukarıdan bakabilme imkanı sağlar.

* İYİLEŞTİRME (enhancement)

Soyutlama düzeyinde irdelene bittikten sonra, tanımlamalarda ayrıntı, düzeltme yapılarak tasarımın daha kesinlik kazanması sağlanır.

* MODÜLERLİK (modularity)

Sistemi istenen kalite faktörleri ışığında parçalara ayrıştırma sonucu elde edilir.

Modülerlik

Bütün karmaşıklığın tek bir modülde toplanması yerine, anlaşılabilir olması için sistemi bir çok modüle ayırır.

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

Veri Tasarımında dikkat edilecek konular

Değişik veri yapıları değerlendirilmeli,
Çok kullanılan veri yapıları için kütüphane oluşturulmalı,
Kullanılacak programlama dili soyut veri tiplerini desteklenmelidir.

Yapısal Tasarım

Yapısal tasarımın amaç hedefi modüler bir yapı oluşturup modüller arasındaki kontrol ilişkilerini temsil etmektir.

Tasarım Kalite Ölçütleri

- Bağlaşım (Coupling)
  Tasarımı oluşturan modüller arası ilişki ile ilgilidir.

- Yapışıklık (Cohesion)
  Modüllerin iç yapısı ile ilgilidir.

BAĞLAŞIM

- Modüller arası bağlılığın ölçülmesi için kullanılan bir ölçüttür.
- Yüksek kaliteli bir tasarımda bağlaşım ölçümü az olmalıdır.
- Bağlaşımın düşük olması
  * Hatanın dalgasal yayılma özelliğinin azaltılması
  * Modüllerin bakım kolaylığı
  * Modüller arası ilişkilerde karmaşıklığın azaltılması

  nedenleri ile istenmektedir.

Yalın Veri Bağlaşımı

Herhangi iki modül arasında iletişim yalın verilerle yapılıyorsa //

Karmaşık Veri Bağlaşımı

Herhangi iki modül arasında iletişimde kullanılan parametreler karmaşık veri yapısındaysa

Denetim Bağlaşımı

İki modül arasında iletişim parametresi olarak denetim verisi kullanılıyorsa

İşlevsel Bağımsızlık

Modüllere parametre ile veri gönderilir ve sonuç değere alınır. Bu modülü çağıran program parçası sadece bu sonucu kullanabilir. Çağrılan modülün işlevsel olarak yaptıkları ile ilgili değildir.

Ortak Veri Bağlaşımı

Eğer iki modül ortak bir alanda tanımlanmış verilere ulaşabiliyorsa bu iki modül ortak veri bağlaşımı olarak tanılar

Zorlukları
* Ortak veri alanını izlemek zordur.
* Ortak veri kullanan modüllerde yapılan değişiklikler diğer modülleri etkiler
* Ortak veri üzerinde yapılacak değişikliklerde bu veriyi kullanarak bütün modüller göz önüne alınmalıdır.

YAPIŞIKLIK

- Bir modülün kendi içindeki işlemler arasındaki ilişkilere ilişkin bir özelliktir. Modül gücü olarakta tanımlanır.
- Tasarımda yapışıklık özelliğinin yüksek olması tercih edilir.
- Yapışık ile Bağlaşım ters orantılıdır.

İşlevsel Yapışıklık

İşlevsel yapışık bir modül, tek bir iş problemine ilişkin sonucu üreten modül olarak tanımlar.

- Maaş-Hesapla, Alan-Hesapla

Sıralı Yapışıklık

Bir modülün içindeki işlevler incelendiğinde, bir işlemin çıktısı, diğer bir işlemin girdisi olarak kullanılıyorsa bu modül sıralı yapışık bir modül olarak adlandırılır.

- Doğrulama-Kaydı-Onayla

İletişimsel Yapışıklık

Bir modülün içindeki farklı işlevler aynı girdi ya da çıktıyı kullanıyorsa bu modül iletişimsel yapışık

Maaş-Bilgisini-Ver

Yordamsal Yapışıklık

Yordamsal yapışık modüldeki işlemler arasında denetim ilişkisi bulunmaktadır.

İşlemlerin birbirleri ile veri ilişkisi yoktur, ancak işlev sırası önemlidir

Zamansal Yapışıklık

Bir modül içindeki işlemlerin sırası önemli değilse, zamansal yapışıklık.

Mantıksal Yapışıklık

Mantıksal olarak aynı türdeki işlemlerin bir araya toplandığı modüller mantıksal yapışıklık olarak adlandırılır

Gelişigüzel Yapışıklık

İşlemler arasında herhangi bir ilişki bulunmaz

Bölüm - 6 / GERÇEKLEŞTİRİM

- Tasarım sonucu üretilen süreç ve veritabanının fiziksel yapısını içeren fiziksel modelin bilgisayar ortamında çalışan yazılım biçimine dönüştürülmesi çalışmasıdır.
- Her şeyden önce bir yazılım geliştirme ortamı seçilmelidir.

VTYS dosya yapısından farkı

~~* Katalog bilgisi, Veri sözlüğü varlığı~~
* Veri soyutlama
* Program işlem bağımsızlığı
* Birden çok kullanıcı desteği
* Birden fazla işlem arası paylaşım

Veri Modelleri

- Fiziksel veri modelleri verinin bilgisayarda nasıl saklandığının detaylarıyla ilgili kavramlar sağlar.

VTYS Mimarisi

Üç seviyede tanımlanabilir.

1- Fiziksel Düzey: Veritabanının fiziksel saklama yapısı
2- Kavramsal Düzey: Kavramsal şema, veritabanının yapısı
3- Dışsal Düzey: Kullanıcı görüşleri

VTYS'nin Sınıflandırılması

Sınıflandırmalar kullanılan veri modeline göre yapılır.

- İlişkisel model: Her bir tablo bir dosya olarak saklanabilir.

- AP Modeli: Veri kayıt

+ Hiyerarşik Model: Veri ağaç yapısı

- Nesne Yönelimli Model:

CASE Araç ve Ortamları

Günümüzde bilgisayar destekli yazılım geliştirme ortamları oldukça gelişmiştir.

CASE araçları yazılım geliştirme sürecinin her aşamasında üretilen bilgi ya da belgelerin bilgisayar ortamında saklanmasına ve bu yolla kolay erişilebilir ve yönetilebilir olmasına olanak sağlar.

* Belirli bir düzeyde kodlama yazılımın yaşam döngüsü açısından önem kazanmaktadır.

Etkin Kod Yazılım Stili Yöntemleri

- Açıklama Satırları
- Kod Yazım düzeni
- Anlamlı İsimlendirme
- Yapısal Programlama Yapıları

Bölüm-7 / YAZILIM DOĞRULAMA VE GEÇERLEME

- Geliştirilecek bilgi sistemi yazılımının doğrulaması
- Ürünün teknik yeterliliğinin değerlendirilmesi
- Geliştirilen belirtimlerin gerçek belirtimlerle karşılaştırılması

Doğrulama vs Geçerleme

- Doğrulama : Doğru ürünü mü üretiyoruz ?
- Geçerleme : Ürünü doğru mu üretiyoruz.

* **Doğrulama** ürünü kullanacak kişilerin isteklerinin karşılayıp karşılamadığını test eden etkinliklerden,

* **Geçerleme** ise ürünün içsel niteliğine ilişkin izleme ve denetim etkinliklerinden oluşur.

Sınama Kavramları

Sınama İşlemleri

- Birim Sınama
- Alt-sistem sınama
- Sistem sınama
- Kabul Sınaması

Alfa ve Beta Sınaması

Alfa Sınamada ; sistemin geliştirildiği yerde kullanıcıların geterek katkıda bulunması sistemi test etmesi amaçlanmakta.

Beta Sınamada ; kullanıcı, geliştirilen sistemi kendi yerleşkesinde, bir gözetmen eşliğinde yapar.

* Sınamalar, hatalardan kurtulmanın bir güvencesi değildir.
* Yazılımın kritiklik düzeyine göre sınamaya ayrılan süre ve çaba artar.

Doğrulama ve Geçerleme Yaşam Döngüsü

- Gerçekleştirim aşamasına kadar olan süreçlerde doğrulama ve geçerleme işlemlerinin planlaması yapılır.
  Planlama paralellikle;
  - alt sistem
  - bütünleştirme
  - sistem ve
  - kabul sınamalarının

  tasarımlarını içerir.

- Gerçekleştirim aşamasının sonunda ise söz konusu plan uygulanır.

SINAMA YÖNTEMLERİ

- Her yazılım mühendisliği ürünü iki yoldan sınanır;

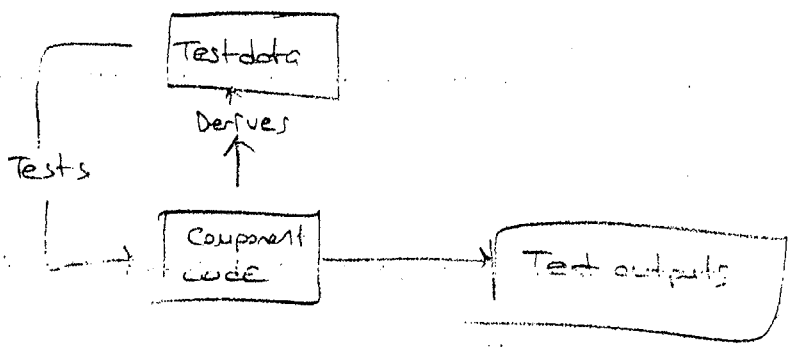
KARAKUTU TESTİ (BLACK-BOX TESTING)

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

BEYAZKUTU TESTİ

İç işlemlerin belirtimlere uygun olarak yürütüldüğünün bileşenler tabanda sınanmasıdır.

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



Sınama ve Bütünleştirme Stratejileri

- Genellikle sınama stratejisi, bütünleştirme stratejisi ile birlikte değerlendir.
- Ancak bazı sınama stratejileri bütünleştirme dışındaki hataları hedefleyebilir.
- Örneğin, yukarıdan-aşağı ve aşağıdan yukarı stratejileri bütünleştirme yöntemine bağlı

Yukarıdan Aşağıya Bütünleştirme

- Önce sistemin üst düzeylerinin sınaması, sonra aşağıya doğru dal düzeylere inilip sınaması
- En üst noktadaki bileşen sınandıktan sonra alt düzeye geçilmelidir.
- Alt bileşenler henüz hazırlanmamışlardır. Bu sebeple kocanlar kullanılır.
  Kocan: Bir alt bileşenin, üst bileşen ile arayüzünü temin eden, fakat işlevsel olarak hiçbir şey yapmayan çerçeve programdır.

İki Temel Yaklaşım

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.

Aşağıdan Yukarıya Bütünleştirme

- Önceki yöntemin tersine uygulama yapılır.
- Bu kez kodlama, bütünleştirme ve sınama, aşağı düzeylerden yukarı düzeylere doğru gelir.
- Önce en alt düzeydeki işçi birimler sınanır ve bir üst düzey ile sınaması gerektiğinde bu düzey bir sürücü ile temsil edilir.

Yaşam Döngüsü Boyunca Sınama

- Planlama ⟶ Sistem sınama planı
- Çözümleme ⟶ Alt sistem sınama planı
- Tasarım ⟶ Modül Sınama Planı
- Gerçekleştirim ⟶ Bütünleştirici Sınama
- Kurulum ⟶ Kullanıcı Sınaması