Dr. MUHAMMET BAYKARA
Fırat Üniversitesi Teknoloji Fakültesi
Yazılım Mühendisliği Bölümü

Sunum İçeriği

- Tanışma / Ben kimim?
- Yazılım Mühendisliği
- Bölüm Hakkında Temel Yönlendirmeler
- Sektörel Beklentiler
- Zorunlu Yaz Stajları
- İş Yeri Eğitimi
- Proje Dökümantasyonu/Rapor
- Öneriler

Ben Kimim?

- Dr. Muhammet BAYKARA
  - Fırat Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü- *Lisans*
  - Fırat Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü- *Yüksek Lisans*
  - Fırat Üniversitesi Teknoloji Fakültesi Yazılım Mühendisliği Bölümü- *Doktora*
  - Fırat Üniversitesi Teknoloji Fakültesi Yazılım Mühendisliği Bölümü- Öğretim Üyesi-Bölüm Başkan Yardımcısı
  - Netix Bilişim Teknolojileri- *Danışman*
  - Smartbee Bilişim Teknolojileri- *Danışman*
  - **Raven -*Danışman***
  - **Innovance – *Danışman***
  - **Crafttech- *Danışman***
  - ***dataSpecta- Danışman***
  - Logical Defense Yazılım Ar-ge Savunma Sanayi Ltd. Şti.- *Kurucu Ortak*
  - *Paygate Yazılım-Danışman*

Ben Kimim?

- Yazılım Mühendisliği Bölümü- Bölüm Başkan Yardımcısı
- Eğitim Öğretim Komisyonu Üyesi
- İşyeri Eğitimi Komisyon Başkanı
- Fen Bilimleri Enstitüsü – Kalite Komisyonu Üyesi
- Verdiğim Derslerden Bazıları:
  - Bilgisayar Bilimlerine Giriş,
  - Yazılım Mühendisliğinin Temelleri / Yazılım Proje Yönetimi
  - Yazılım Tasarım ve Mimarisi
  - Bilgi Sistemleri ve Güvenliği / Siber Savaş ve Terörizm/İleri Ağ Güvenliği
  - İnsan Bilgisayar Etkileşimi
  - Yazılım Kalite Güvencesi ve Testi

Ben Kimim?

- İletişim Bilgilerim,
  - www.muhammetbaykara.com
  - mbaykara@firat.edu.tr,
  - 0 424 237 00 00-4296
  - Ofis Saati : Pazartesi 13:00-15:00

Yazılım Mühendisliği

- Nisbi olarak daha yeni
- Standartları güncel (SQAP&SVVP, aft 2002)
- Bilgisayar Mühendisi (Computer Eng.)
- Yazılım Mühendisi (Software Eng.)
- Yazılım Mimarı (Software Architecture)

Bölümümüz

- 2009
- Danışman
- Bölüm Başkan Yardımcısı
- Bölüm Başkanı
- Akademik Kadro (Öğretim Üyesi, Öğretim Görevlisi, Araştırma Görevlisi)
- İdari Kadro (Bölüm Sekreteri)

Bölümümüz

- Akademik (danışma)
  - İbrahim Türkoğlu& Bihter Daş& Cem Baydoğan
- İdari (Öğrenci işleri, dilekçeler, başvurular, genel danışma)
  - Şahin Çelikten& Ersin Ercan
- Öğrenci Görüşme Saati İsteyiniz! (2 Saat/Hafta) / 1500!!!
- Hiyerarşi
  - Danışman, Bölüm Baş. Yrd., Bölüm Başkanı
- Komisyonlar
- https://yazilimtf.firat.edu.tr/

Dersler

1. Yarıyıl Ders Planı					
Kodu	**Dersin adı**	**T**	**U**	**K**	**AKTS**
FIZ111	Fizik I	4	0	4	6
FİZ105	Fizik Lab. I	0	2	1	2
MAT161	Matematik I	4	0	4	6
TRD109	Türk Dili I	2	0	2	2
YDI107	İngilizce I	2	0	2	2
YMH111	Algoritma ve Prog. I	3	2	4	6
YMH113	Bilgisayar Bilimlerine Giriş	3	0	3	4
YMH115	Yazılım Mühendisliği Oryantasyonu	0	2	1	2
	Toplam Kredi	**18**	**6**	**21**	**30**

2. Yarıyıl Ders Planı					
Kodu	**Dersin adı**	**T**	**U**	**K**	**AKTS**
FIZ112	Fizik II	3	0	3	5
FİZ106	Fizik Lab. II	0	2	1	2
MAT162	Matematik II	4	0	4	6
TRD110	Türk Dili II	2	0	2	2
YDI108	İngilizce II	2	0	2	2
YMH112	Algoritma ve Prog. II	3	2	4	7
YMH114	Yazılım Müh. Temelleri	3	0	3	6
	Toplam Kredi	**17**	**4**	**19**	**30**

3. Yarıyıl Ders Planı					
Kodu	**Dersin adı**	**T**	**U**	**K**	**AKTS**
AIT209	Atatürk İlk. ve İnk. Tar I	2	0	2	2
İST215	Olasılık ve İstatistik	3	0	3	4
MAT215	Lineer Cebir	2	0	2	4
YMH211	Ayrık Yapılar	3	0	3	5
YMH213	Mesleki İngilizce	2	0	2	3
YMH215	Sayısal Tasarım	3	2	4	6
YMH217	Nesne Tabanlı Programlama	3	2	4	6
	Toplam Kredi	**18**	**4**	**20**	**30**

4. Yarıyıl Ders Planı					
Kodu	**Dersin adı**	**T**	**U**	**K**	**AKTS**
AIT210	Atatürk İlk. ve İnk. Tar. II	2	0	2	2
MAT220	Diferansiyel Denklemler	4	0	4	6
YMH210	İngilizce İletişim Becerileri	2	0	2	3
YMH212	Yazılım Gereksinimleri ve Analizi	3	2	4	5
YMH214	Sayısal Analiz	2	2	3	4
YMH216	Yazılım Ekonomisi	2	0	2	3
YMH218	Veri Yapıları	3	0	3	4
YMH220	İleri Programlama Teknikleri	2	0	2	3
	Toplam Kredi	**20**	**4**	**22**	**30**

Dersler

5. Yarıyıl Ders Planı					
Kodu	**Dersin adı**	**T**	**U**	**K**	**AKTS**
YMH311	Yazılım Tasarım ve Mimarisi	2	2	3	3
YMH313	İşletim Sistemleri	3	2	4	4
YMH315	Veritabanı Yönetim Sis.	3	2	4	5
YMH317	Algoritma Analizi	3	0	3	3
YMH319	Programlama Dilleri	3	0	3	3
	SEÇMELİ DERSLER	6	4	8	12
	Toplam Kredi	**20**	**10**	**25**	**30**

5. Yarıyıl Seçmeli Ders Listesi					
Kodu	**Dersin adı**	**T**	**U**	**K**	**AKTS**
YMH321	Bilgi Sistemleri ve Güv.	3	2	4	6
YMH323	Ağ Sistemleri	3	2	4	6
YMH325	Mikroişlemciler ve Prog.	3	2	4	6
YMH327	Mobil Programlama	3	2	4	6
YMH329	İnsan Bilgisayar Etkileşimi	3	2	4	6
YMH331	Dağıtık Yazılım Mühendisliği	3	2	4	6
YMH333	Paralel Programlama	3	2	4	6
YMH335	Grafik Programlama	3	2	4	6

6. Yarıyıl Ders Planı					
Kodu	**Dersin adı**	**T**	**U**	**K**	**AKTS**
YMH310	Web Tasarımı ve Prog.	3	2	4	5
YMH312	Biçimsel Dil. ve Oto. Teo.	3	2	4	5
YMH314	Fonksiyonel Programlama	3	2	4	5
YMH316	İş Hukuku	2	0	2	3
	SEÇMELİ DERSLER	6	4	8	12
	Toplam Kredi	**17**	**10**	**22**	**30**

6. Yarıyıl Seçmeli Ders Listesi					
Kodu	**Dersin adı**	**T**	**U**	**K**	**AKTS**
YMH318	Optimizasyon Teknikleri	3	2	4	6
YMH320	Bilgisayarlı Grafik ve Ani.	3	2	4	6
YMH322	Veri Madenciliği	3	2	4	6
YMH324	Çoklu Ortam Yazılımı Geliş.	3	2	4	6
YMH326	Biyoinformatik	3	2	4	6
YMH328	Veritabanı Tas. ve Yön.	3	2	4	6
YMH330	Java Programlama	3	2	4	6
YMH332	Ağ Programlama	3	2	4	6

Dersler

7. Yarıyıl Ders Planı					
Kodu	**Dersin adı**	**T**	**U**	**K**	**AKTS**
YMH411	İş Yeri Eğitimi	5	0	5	5
YMH413	İş Yeri Uygulaması	0	15	8	15
YMH415	Mesleki Uygulama-I	0	2	1	5
YMH417	Mesleki Uygulama-II	0	2	1	5
	Toplam Kredi	**5**	**19**	**15**	**30**

8. Yarıyıl Ders Planı					
Kodu	**Dersin adı**	**T**	**U**	**K**	**AKTS**
YMH412	Yazılım Kalite Güven. ve T.	3	2	4	6
YMH414	Bitirme Projesi	0	2	1	3
YMH416	İnsan Kaynakları Yönetimi	2	0	2	3
YMH418	Yazılım Müh. Güncel Konular	3	2	4	6
	SEÇMELİ DERSLER	6	4	8	12
	Toplam Kredi	**14**	**10**	**19**	**30**

8. Yarıyıl Seçmeli Ders Listesi					
Kodu	**Dersin adı**	**T**	**U**	**K**	**AKTS**
YMH420	Yapay Zeka ve Uzman Sis.	3	2	4	6
YMH422	Oyun Tasarımı ve Prog.	3	2	4	6
YMH424	Hareketli Yazılım Geliştirme	3	2	4	6
YMH426	Sayısal Görüntü İşleme Yön.	3	2	4	6
YMH428	Robot Programlama	3	2	4	6
YMH430	E-Ticaret	3	2	4	6

Dersler

- Algoritma ve Programlama
- Nesne Tabanlı Programlama
- Veri Yapıları
- Web Programlama&E-Ticaret
- Bilgi Güvenliği-İşletim Sistemleri-Bilgisayar Ağ Sistemleri
- Algoritma Analizi
- Veritabanı Yön. Sis.
- Yazılım Tasarım ve Mimarisi
- Mesleki Uygulama I
- Mesleki Uygulama II
- İş Yeri Eğitimi
- Bitirme

Bazı Tavsiyeler

- ***<u>Yanlışlardan Kaçın(ın)!!!</u>***
- Tekrar(cı) öğrenci yıldırmalarından!
- Sosyal medya araçları üzerinde şehir, üniversite, bölüm ve hocalar hakkındaki iftira, yalan-yanlış, tehdit vb. söylemlerden!
- Yazılım ekip işidir! Ekibi bozan öğrenci tiplerinden!
- Bilişim suçu işlemekten!
- 5651, 6698
- Yanlış sorulardan!
- İş bulabilecek miyiz? Nerde iş bulacağız! gibi vesveselerden…

Bazı Tavsiyeler-Sınavlar

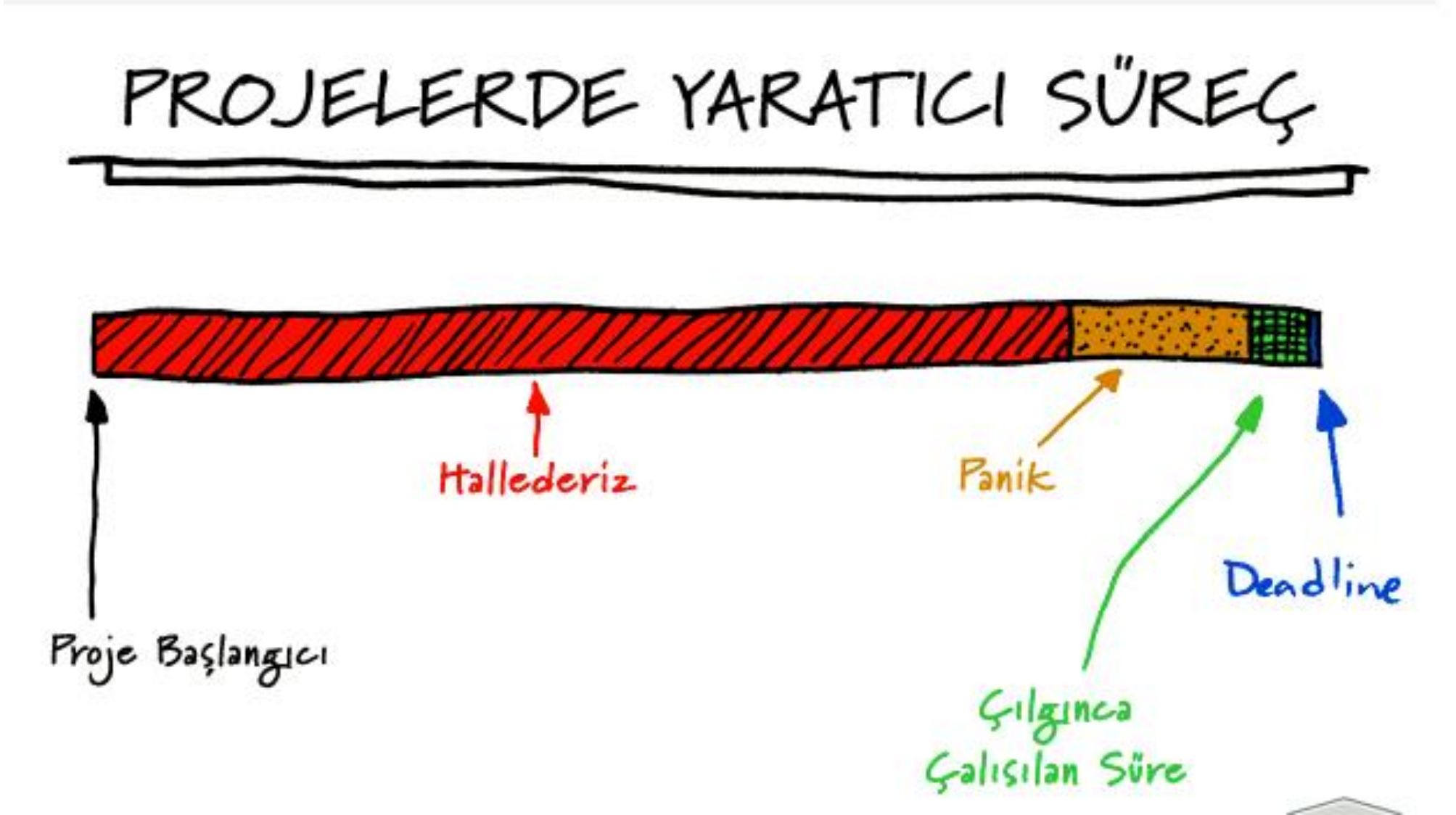
- Sınavlarla ilgili derslik kapılarında dekanlıkça yapılan duyurulara uyulmalıdır.
- Sınavlara cep telefonu ile gelinmemelidir. Aksi halde Tutanak tutulur!
- Sınavlar ilgili öğretim üyesinin duyuracağı biçimde yapılır.
- Sınav süresince gözetmen uyarılarına uyulmalıdır.
- Sınav sonuçları pano veya OBS üzerinden görülür.
- Sonuçlara itiraz durumunda, Bölüm sekreterliğine dilekçe verilebilir.
- Sınıf geçme için 1.8, mezuniyet için 2.0 Ort. Gereklidir.

Bazı Tavsiyeler- Zaman

- **Zamanınızı İyi Değerlendirin**



Bazı Tavsiyeler-Zaman



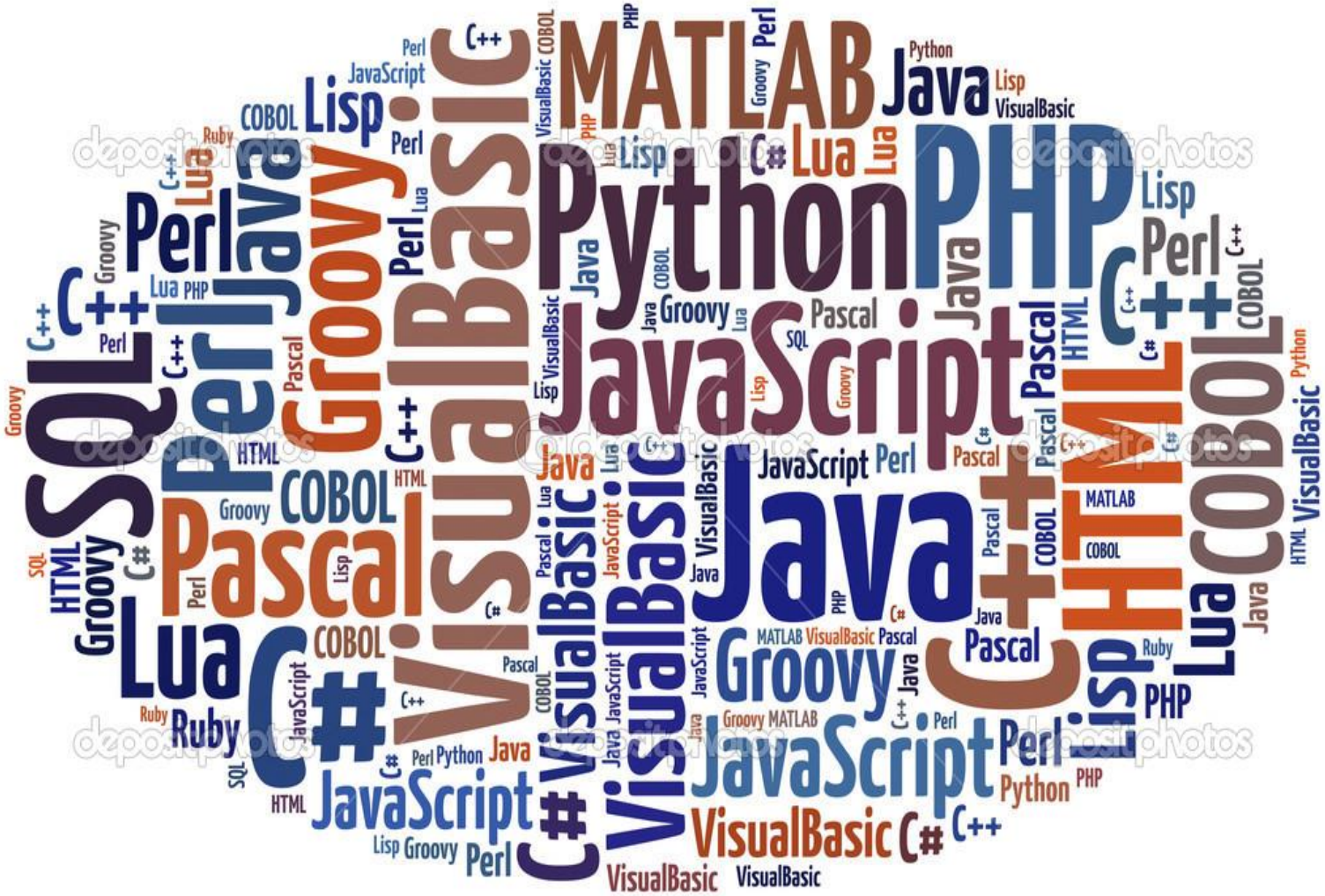
Bazı Tavsiyeler – İngilizce

- Yabancı Dil Öğrenin!!! İngilizce!?
  Yabancı temel kaynaklardan yararlanın



Bazı Tavsiyeler Herşey mi? Bir şey mi?

- Her şeyden bir şey mi; Bir şeyden herşey mi ?



Bazı Tavsiyeler

- Diploma, Ortalama, Üniversite



Eğitim, Etkinlik ve Sertifikalar

- CV, Etkinlik, Kulüpler, Referanslar, Sertifikalar, Blog, Makale, Rapor

BilgeAdam BİLGİSAYAR TEKNOLOJİLERİ AKADEMİSİ

Microsoft®



ISO

ISO/IEC 27001



INNOVANCE
Innovation in Advance

BGA SECURITY

Bazı Tavsiyeler

Ekip halinde çalışmanın önemi;
Fikir Çimlenmesi & Beyin Fırtınası





Bazı Tavsiyeler

- I. Kısım Yaz Stajı- 2. Sınıf Sonu- 20 İş Günü
- II. Kısım Yaz Stajı-3. Sınıf Sonu- 20 İş Günü

- Staj ile ilgili komisyon
  - Bihter Daş
- Rapor oluşturma-Günlük raporlama
- Defterin hazırlanması (Format, kapak, şablon, paraf, imza)
- Gizli Evrak, Sigorta,
- Süreç için klavuz ve akış şeması ()

Bazı Tavsiyeler

- 7. ya da 8. yarıyıl
- 14 haftalık işyerinde staj eğitimi

- İşyeri eğitimi ile ilgili komisyon
  - Resul Daş, Muhammet Baykara, Yaman Akbulut, Alev Kaya
- Rapor oluşturma-Haftalık veya proje bazlı raporlama
- Defterin hazırlanması (Format, kapak, şablon, paraf, imza)
- Gizli Evrak, Sigorta,
- Detaylı bilgi için (Bölüm Web Sayfası)

Raporlama&Dökümantasyon

- Yazılım Mühendisliğinin Temelleri,
- Yazılım Gereksinimleri ve Analizi
- Yazılım Tasarım ve Mimarisi,
- Ödevler
- Bitirme Projesi
- Staj I, Staj II, İş yeri eğitimi

Oehri ve Teufel tarafından yapılan çalışmada, sosyal ağlarda güvenlik konusu kullanıcılar bağlamında ele alınmıştır. Şirketlerin sosyal ağlardaki güvenlik kaygıları incelenmiş ve güvenlik zinciri halkalarının zayıflık dereceleri gösterilmiştir. Bu verilerden yola çıkarak, güvenlik sorununun en önemli sebebinin sosyal ağların bilinçsiz kullanımından kaynaklandığı vurgulanmıştır. Çalışanlar, paylaşımlarına dikkat etmedikleri takdirde çok büyük güvenlik açıklarının oluşabileceği anlatılmış ve bir önlem olarak kullanıcıların sosyal ağ kullanımında dikkatli olmaları gerektiği vurgulanmıştır [2]. Chen ve Shi tarafından

KAYNAKLAR

[1] Yıldırım N., Varol A., "Sosyal Ağlarda Güvenlik: Bitlis Eren ve Fırat Üniversite'lerinde Gerçekleştirilen Bir Alan Çalışması," 1st International Symposium on Digital Forensics and Security (ISDFS'13), Elazığ, 2013.

[2] Oehri C., Teufel S., " Social media security culture," *Information* Security for South Africa (ISSA), Johannesburg, 2012.

Proje Destekleri

- TGSD-1512
- Kosgeb
- Kalkınma Ajansları-FKA
- Teknokent
- TUBİTAK- 2209, Lisans Öğr. Yaz. Yarışmaları
- FUBAP Lisans Öğrencileri Araştırma Projesi (6-12 ay, 2.5 Ort.)

Yazılım Koçluğu-Danışmanlık Programı

- Nedir?
- Ne içerir? Ne Sağlar?
- Kimler katılmalı?
- Ne zaman kod yazabileceğim!?
- İşleyiş Nasıl?
- Amaç nedir?

Yazılım Koçluğu-Danışmanlık Programı

Nedir?

- Inspimo firmamızın, Innovance, Raven, Craft gibi çözüm ortakları ile birlikte geliştirdiği bir kariyer planlama programıdır.
- Yazılım mühendisliği öğrencileri için özel bir programdır.
- Öğrenciler başvurular arasından mülakat ile seçilir.
- Frontend, backend ve full stack developer yetiştirmeyi amaçlar.
- Dil bağımsız bir mantıktan yapay zeka ve makine öğrenmesine kadar geniş bir yelpaze sunar.
- Bölüm derslerine mutlak yardım sağlar.
- Mühendis adayını henüz okurken hayata hazırlar.
- İstihdam garantili özel eğitim kampları içerir.

Yazılım Koçluğu-Danışmanlık Programı

Ne İçerir / Ne Sağlar?

- Özel çalışma ortamı, teknokent ofislerinde çalışma imkanı sağlanır.
- CV'niz gerçek projeler ile doldurulmaya başlanır.
- CV'nize akademik ve iş hayatından gerçek proje bazlı referans olunur.
- Gerçek projelerde çalışma imkanı sağlanır. Detaylar ofis ziyaretinde.
- Ekip çalışmaları ile iş hayatına hazırlık sağlanır.
- Mesleki Uygulama I, Mesleki Uygulama II ve İşyeri Eğitimi imkânı sağlanır.
- Algoritma, Yazılım Modelleme, Java, Android, Python, Unity, Unreal Engine, Photoshop, React, Swift, Arduino, Lora, PHP, CMS/WP, İngilizce…

Yazılım Koçluğu-Danışmanlık Programı

- 72 Kurs
- 5484 Tane ders
- 705 Saat 56 Dakika
- Kütüphane
- Dijital Kütüphane- son üç yıl basımlı sınırsız dijital arşiv
- 3 Ofis (dataSpecta, Innovance, Crafttech)

Yazılım Koçluğu-Danışmanlık Programı

Temel Asp.Net Web Api	Temel Düzeyde Asp Net Web Api ve Jquery ile Get ve Post İşlemleri	6 ders 51:37	AOS SOFTWARE	.NET	Türkçe
Komple ASP.NET Web Geliştirme Eğitimi	Web Development: HTML,CSS,Bootstrap 4,JavaScript ES6+,Angular,C#,Entity Framework,ASP.NET MVC & CORE,API,SQL Veri Tabanı	516 ders 62:00:54	Engin Demiroğ	.NET	Türkçe
Visual Studio İpuçları& .NET Temelleri Eğitim Videosu Serisi	Visual Studio Geliştirme Ortamını (IDE) Tanıyoruz. [Yazılım]	28 ders 03:00:08	Alper ÇELİK	.NET	Türkçe
Yeni Başlayanlar İçin Adobe Dreamweaver Temel Kod Eğitimi	Bu eğitimle sektördeki poüler yazılımları öğrenerek web tasarım, grafik ve animasyon yeteklerinizi geliştirebilirsiniz.	22 ders 03:30:11	Fatih Gök	Dreamweaver	Türkçe
Yeni Başlayanlar İçin Adobe Flash Temel Animasyon Eğitimi	Bu eğitimle sektördeki popüler yazılımları öğrenerek web tasarım, grafik ve animasyon yeteklerinizi geliştirebilirsiniz.	28 ders 02:57:07	Fatih Gök	Flash	Türkçe
Yeni Başlayanlar İçin Adobe Photoshop Temel Tasarım Eğitimi	Bu eğitimle Photoshop kullanımı öğrenerek fotoğraf,web ve grafik tasarım için yeteneklerinizi geliştirebilirsiniz.	18 ders 01:48:22	Fatih Gök	Photoshop	Türkçe
A'dan Z'ye Angular Kursu | 2019	Google tarafından geliştirilen en popüler front-end framework'ü Angular 'ı sıfırdan en ileri seviyelere kadar öğrenin.	113 ders 16:24:28	Engin Demiroğ	Angular.JS	Türkçe
Sıfırdan Bootstrap 4 Öğrenme	Bootstrap Kullanarak Mobil Uyumlu Websiteleri Yapmak İstiyorsanız Bu Kursu Kaçırmayın	40 ders 04:16:29	Ökkeş Aksoy	Bootstrap	Türkçe
A'dan Z'ye Bootstrap 4	Bootstrap 4'ü baştan sona öğrenerek ekran boyutuna duyarlı web sayfaları oluşturun.	52 ders 02:20:36	Bootstrap Eğitimi	Bootstrap	Türkçe
Baştan sona C Programlama eğitimi	Hızlı ve Kolay Anlaşılır C kursu	52 ders 03:51:32	Tolgahan Özcan	C	Türkçe
C# ile Katmanlı Mimari Öğren : Ado .Net ile WinForm	C# WinForm üzerinde Ado .Net ve Katmanlı mimariyle temel veritabanı işlemleri yapmayı öğrenin.	16 ders 01:22:00	Ercan Ayhan	C#	Türkçe
C# Öğrenin	C#	24 ders 04:33:05	Ömer Faruk Sarıkaya	C#	Türkçe
Bir Saatte Css Öğren	Kısa sürede css kursuyla web tasarımcı olun.	32 ders 01:42:40	Ercan Ayhan	CSS	Türkçe
CSS Web Development Crash Course	Learn how to style awesome websites and web applications for the internet.	27 ders 02:39:12	Frahaan Hussain	CSS	İngilizce
Web Yazılıma Giriş Html Css Javascript Temel Eğitim	Yazılım dünyasına en basit yerden giriş yapmak için bu kursu almalısınız	24 ders 03:22:42	Mehmet Ali BAYRAM	CSS	Türkçe
Electron.js ile Masaüstü Uygulama Yapımına Giriş	Sadece JavaScript kullanarak harika masaüstü uygulamaları yapmak istiyorsanız. Electron.js tam size göre!	21 ders 02:49:04	Gökhan Kandemir	Electron.JS	Türkçe
Game Maker ile Sıfırdan Oyun Yapımı - Bir Oyun Yapımcısı Ol	Oyun yapmayı öğrenerek kariyerinizde yeni bir yol seçin.	17 ders 02:39:58	Altan KAHRAMAN	Game Maker	Türkçe
Programlamaya Giriş : HTML	Temel HTML Bilgisi	7 ders 34:59	SKB SOFTWARE	HTML	Türkçe
Bir Saatte Html Öğren	Html ile kısa sürede web tasarım yapmaya başlayın.	24 ders 01:03:54	Ercan Ayhan	HTML	Türkçe

Yazılım Koçluğu-Danışmanlık Programı

Kimler Katılmalı?

- Kariyerine yazılım mühendisi, yazılım mimarı olarak devam etmek isteyen,
- Azimli, kararlı, sabırlı,
- Problem çözme yeteneği olan,
- Gerçek dünya sorunlarına duyarlı,
- Sahadan analiz yapabilme kabiliyetinde olan,
- Gerçek bir CV oluşturmak isteyen,
- İlgisine göre oluşturulan/önerilen çalışma programına uymayı taahhüt eden,

Yazılım Koçluğu-Danışmanlık Programı

Anahtar kelimelerle yazılım koçluğu programının özeti;

- CV, Özgelecek!
- Kişisel kariyer geliştirme programı,
- Sertifika,
- Eğitim,
- Proje yönlendirmesi,
- Etkinlik, yarışma, hackathon, kamp,
- Ömür boyu oryantasyon
- Staj
- İstihdam
- Danışmanlık

Yazılım Koçluğu-Danışmanlık Programı

İşleyiş nasıl olacak?

- 40 seçilmiş katılımcı, Ekim Teknokent,
- Ortalama 5-10 kişiye bir mühendis danışman atanacak,
- Hafta sonları planlanan kursların belirlenmesi ve duyurulması (min. 8 saat)
- Eğitim dışında mevcut 9-10 projeye göre çalışma takvimi ve iş paketlerinin belirlenmesi.
- Böylelikle sezon sonunda katılımcılar 9-10 gerçek proje referansına sahip olacaktır.
- Ayrıca hafta içinde ve hafta sonu artık zamanlarda danışmanlar ile birebir çalışma imkanı (soru-cevap, yönlendirme vb.)
- Gelişimlere göre süreç sonunda yaz aylarından başlamak üzere, projelere stajer olarak katılım imkanı sağlanarak istihdam adımları atılmış olacak

Yazılım Koçluğu-Danışmanlık Programı

Amaç Nedir?

- Kariyer gelişiminize katkı sağlamak,
- Innovance ve çözüm ortaklarımıza istihdam altyapısı oluşturmak,
- Mühendislik etik kurallarını özümsemiş, kaliteli mühendisler yetiştirmek,
- Üniversitelerdeki güçlü teorik yapıyı pratik ile buluşturmak,
- Birlikte ülkemize milli yazılımlar üretebilmek için kaliteli insan potansiyelini harekete geçirmek,

Soru Cevap?

TEŞEKKÜRLER

Dr. MUHAMMET BAYKARA
Fırat Üniversitesi Teknoloji Fakültesi
Yazılım Mühendisliği Bölümü

Sunum İçeriği

- Arama motorları ve arama teknikleri
- Sektörel Beklentiler / Bilişim Etiği
- Stajlar ve İşyeri Eğitimi
- Proje Dökümantasyonu/Rapor Hazırlama
- Öneriler
- Soru Cevap

Arama motorları ve arama teknikleri

- Google,
- Yandex,
- Bing,
- Baidu,
- DuckDuckGo, SearX, Ecosia, StartPage * Etik
- Ecosia,
- Gigablast,
- Munax,
- Qwant,
- Sogou,
- Yaani * Tr

Akademik Arama Motorları/ *Dijital Kütüphaneler*

- Konferans/Sempozyum → Bildiri
- Dergi→ Makale

- https://www.sciencedirect.com/ Science Direct
- https://ieeexplore.ieee.org/Xplore/home.jsp IEEE XPLORE
- https://scholar.google.com.tr/ Google Akademik
- https://towardsdatascience.com/
- https://paperswithcode.com/
- https://github.com/
- https://www.kaggle.com/

Akademik Arama Motorları/ *Dijital Kütüphaneler*



Akademik Arama Motorları/ *Dijital Kütüphaneler*



sorted by *relevance* | date

Research article

A real-time alert correlation method based on code-books for intrusion detection systems

Computers & Security, Volume 89, February 2020, Article 101661

Ehsan Mahdavi, Ali Fanian, Fatima Amini

Research article

Model of the intrusion detection system based on the integration of spatial-temporal features

Computers & Security, Volume 89, February 2020, Article 101681

Jianwu Zhang, Yu Ling, Xingbing Fu, Xiongkun Yang, ... Rui Zhang

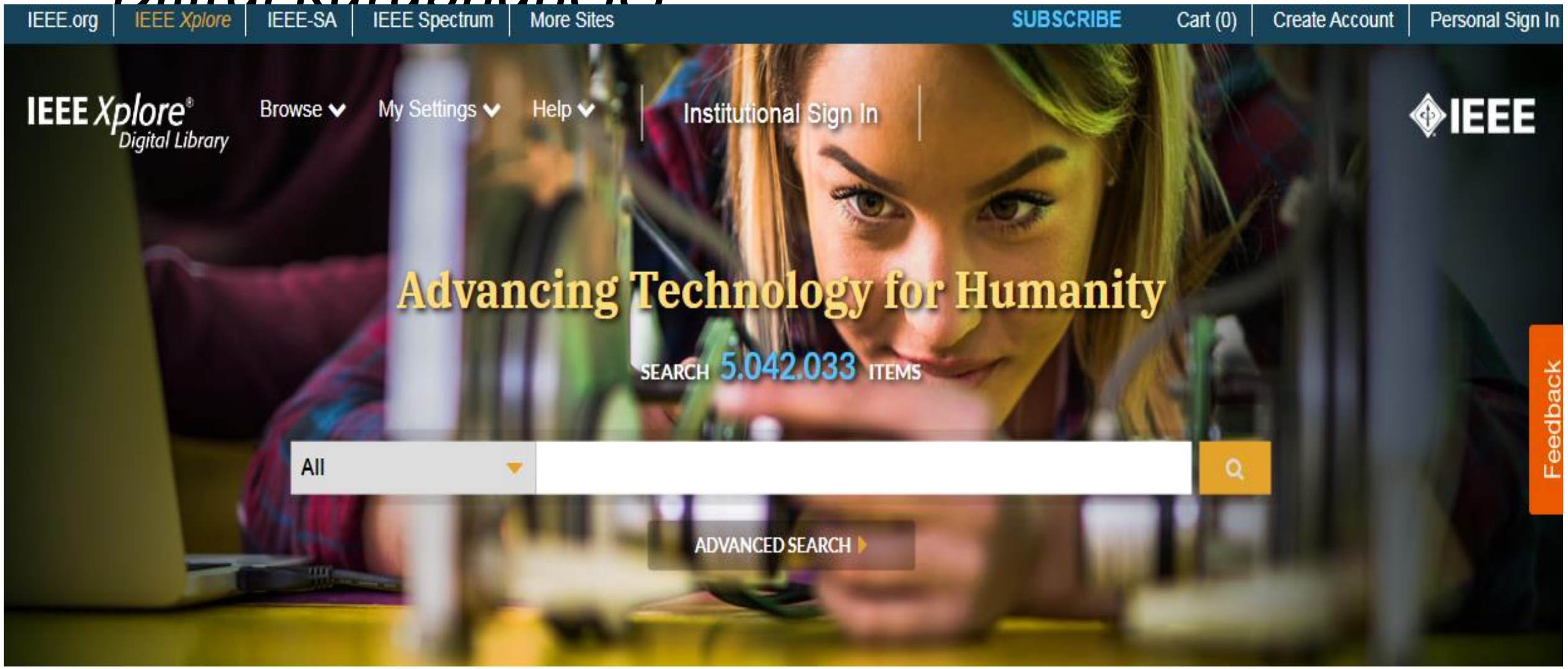
Want a richer search experience?

Sign in for additional filter options, multiple article downloads, and more.

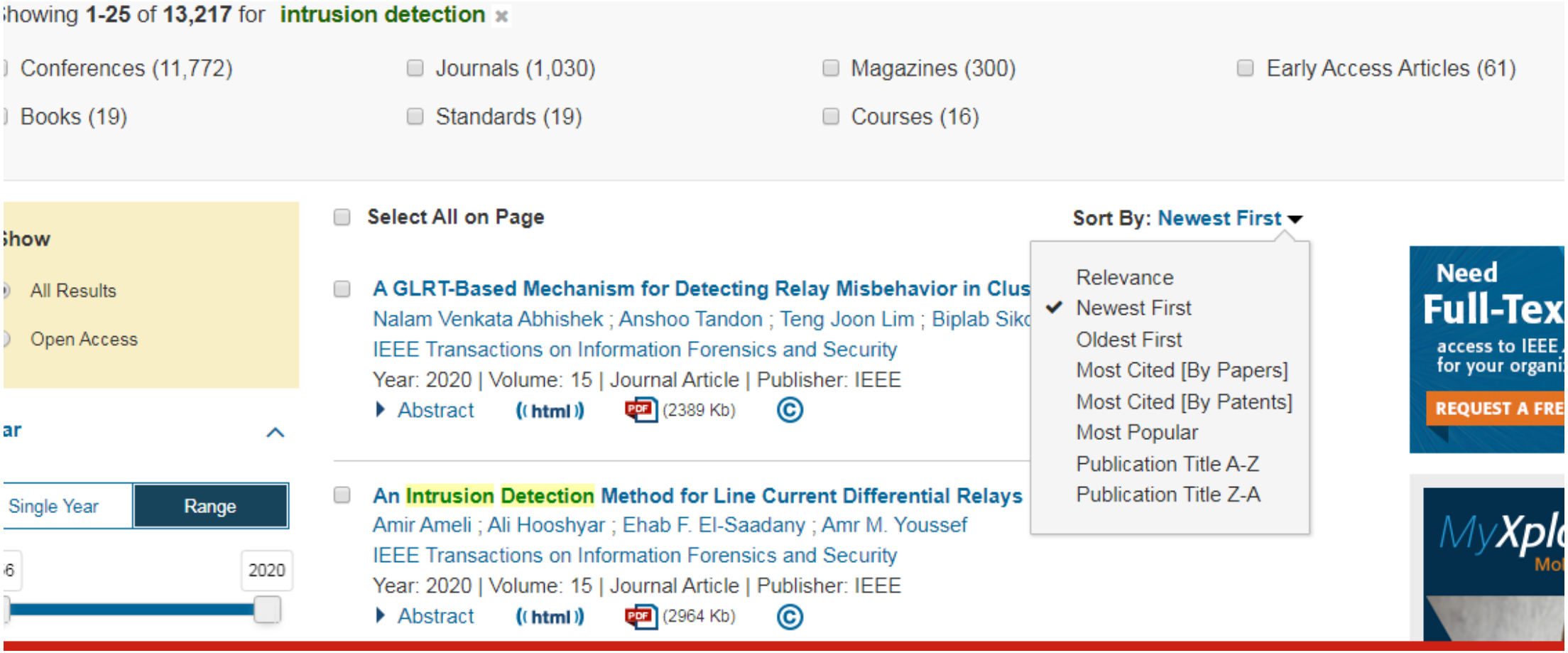
Sign in >

Feedback

Akademik Arama Motorları/ *Dijital Kütüphaneler*



Akademik Arama Motorları/ *Dijital Kütüphaneler*



Akademik Arama Motorları/ *Dijital Kütüphaneler*

Detection of phishing attacks
Muhammet Baykara ; Zahit Ziya Gürel
2018 6th International Symposium on Digital Forensic and Security (ISDFS)
Year: 2018 | Conference Paper | Publisher: IEEE
▸ Abstract ((html)) PDF (534 Kb) ©

Review and comparison of captcha approaches and a new captcha model
Muhammet Baykara ; Furkan Alnıak ; Kahraman Çınar
2018 6th International Symposium on Digital Forensic and Security (ISDFS)
Year: 2018 | Conference Paper | Publisher: IEEE
▸ Abstract PDF (502 Kb) ©

A novel approach to ransomware: Designing a safe zone system
Muhammet Baykara ; Baran Sekin
2018 6th International Symposium on Digital Forensic and Security (ISDFS)
Year: 2018 | Conference Paper | Publisher: IEEE
Cited by: Papers (1)
▸ Abstract ((html)) PDF (846 Kb) ©

Akademik Arama Motorları/ *Dijital Kütüphaneler*

Applications for detecting XSS attacks on different web platforms
Muhammet Baykara ; Sebahattin Güçlü
2018 6th International Symposium on Digital Forensic and Security (ISDFS)
Year: 2018 | Conference Paper | Publisher: IEEE
▸ Abstract (708 Kb) ©

An overview of monitoring tools for real-time cyber-attacks
Muhammet Baykara ; Ugur Gurturk ; Resul Das
2018 6th International Symposium on Digital Forensic and Security (ISDFS)
Year: 2018 | Conference Paper | Publisher: IEEE
▸ Abstract ((html)) (1009 Kb) ©

A review of cloned mobile malware applications for android devices
Muhammet Baykara ; Eren Çolak
2018 6th International Symposium on Digital Forensic and Security (ISDFS)
Year: 2018 | Conference Paper | Publisher: IEEE
▸ Abstract ((html)) (631 Kb) ©

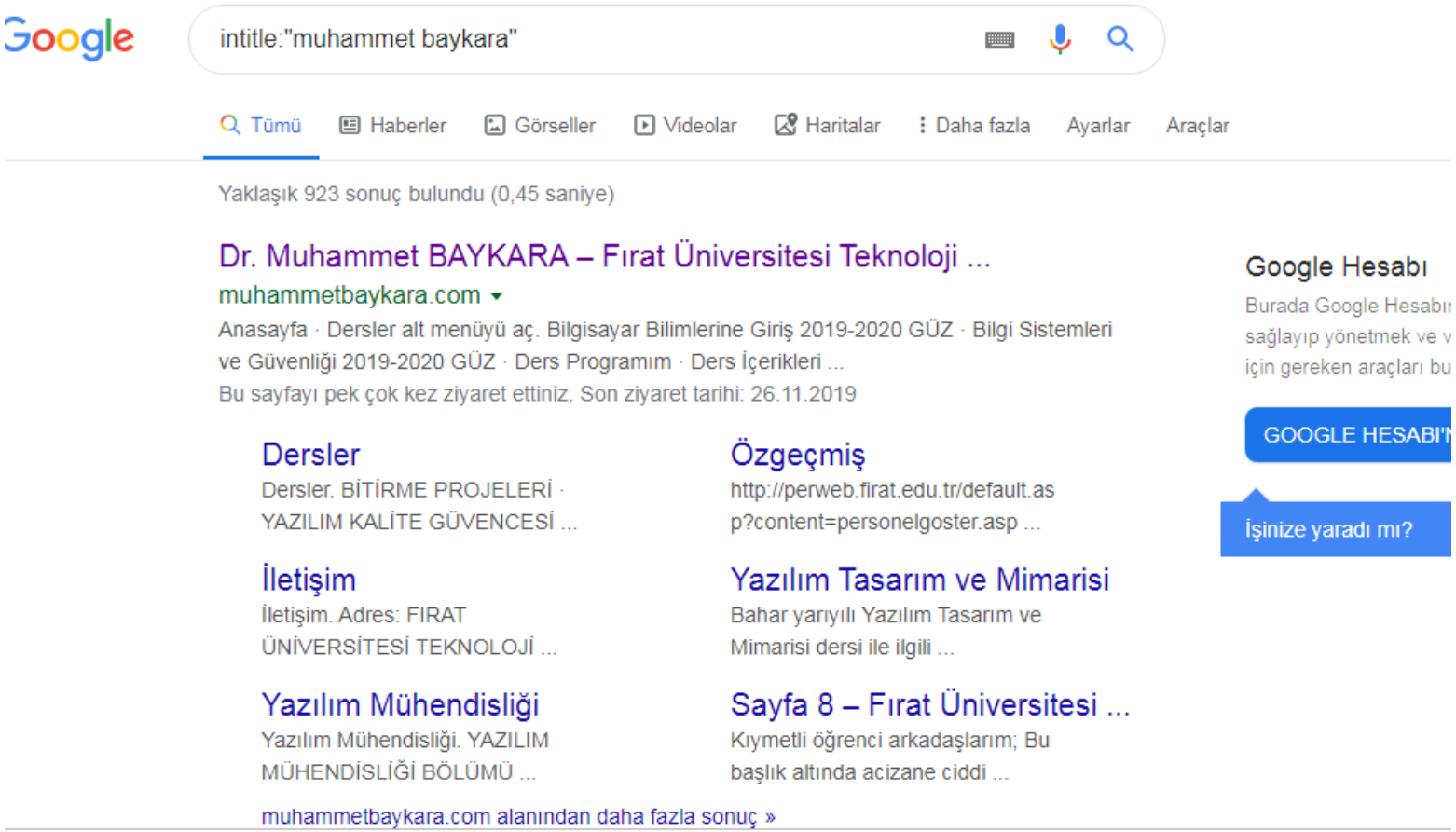
Akademik Arama Motorları/ *Dijital Kütüphaneler*

Augmented reality based mobile learning system design in preschool education
Muhammet Baykara ; Uğur Gürtürk ; Bedirhan Atasoy ; İbrahim Perçin
2017 International Conference on Computer Science and Engineering (UBMK)
Year: 2017 | Conference Paper | Publisher: IEEE
Cited by: Papers (1)
▸ Abstract (416 Kb) ©

Classification of social media shares using sentiment analysis
Muhammet Baykara ; Uğur Gürtürk
2017 International Conference on Computer Science and Engineering (UBMK)
Year: 2017 | Conference Paper | Publisher: IEEE
▸ Abstract (454 Kb) ©

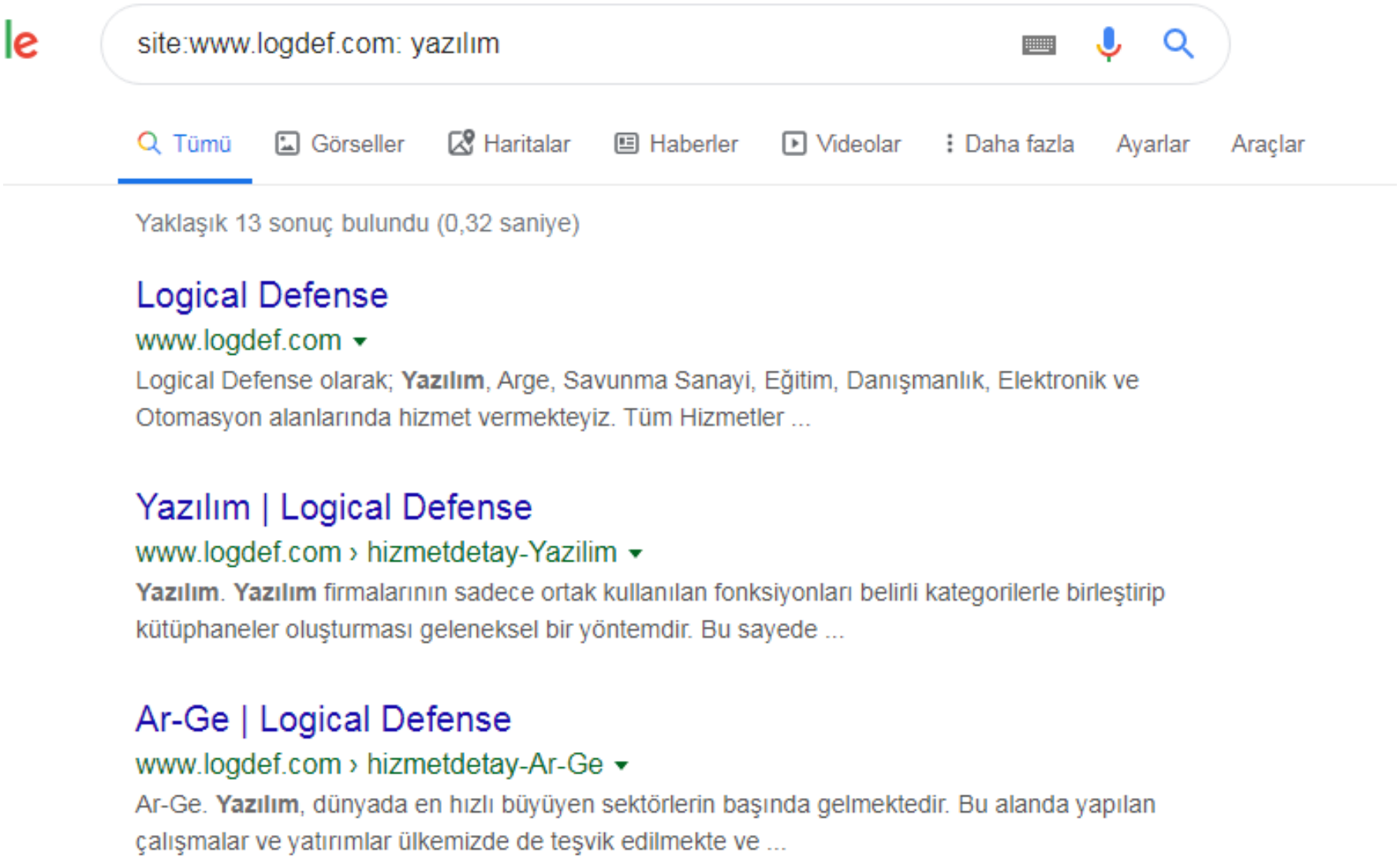
NFC based smart mobile attendance system
Muhammet Baykara ; Uğur Gürtürk ; Ertuğrul Karakaya
2017 International Conference on Computer Science and Engineering (UBMK)
Year: 2017 | Conference Paper | Publisher: IEEE
▸ Abstract (273 Kb) ©

Google ile bazı arama teknikleri
intitle

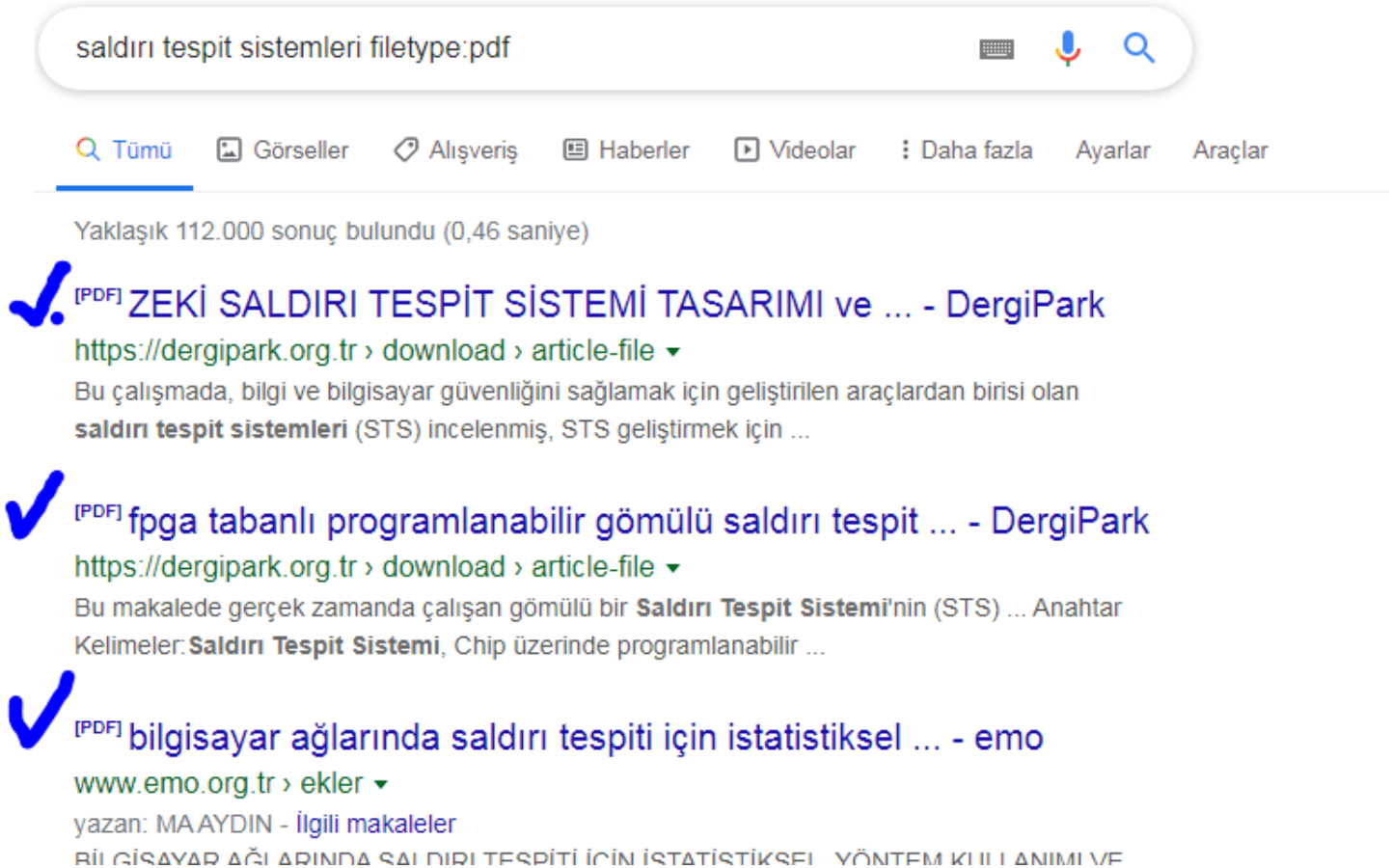


Google ile bazı arama teknikleri

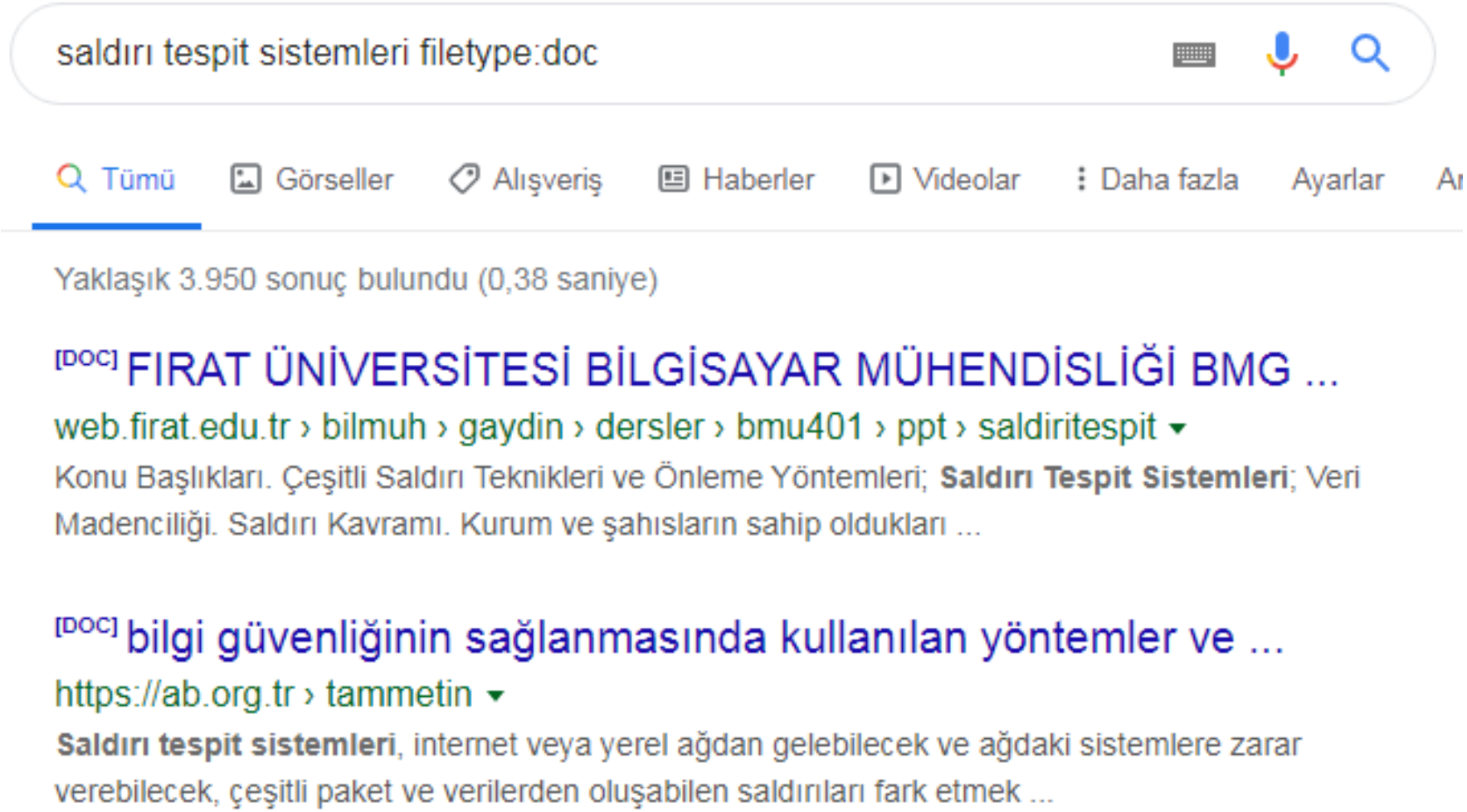
site



Google ile bazı arama teknikleri *filetype*:

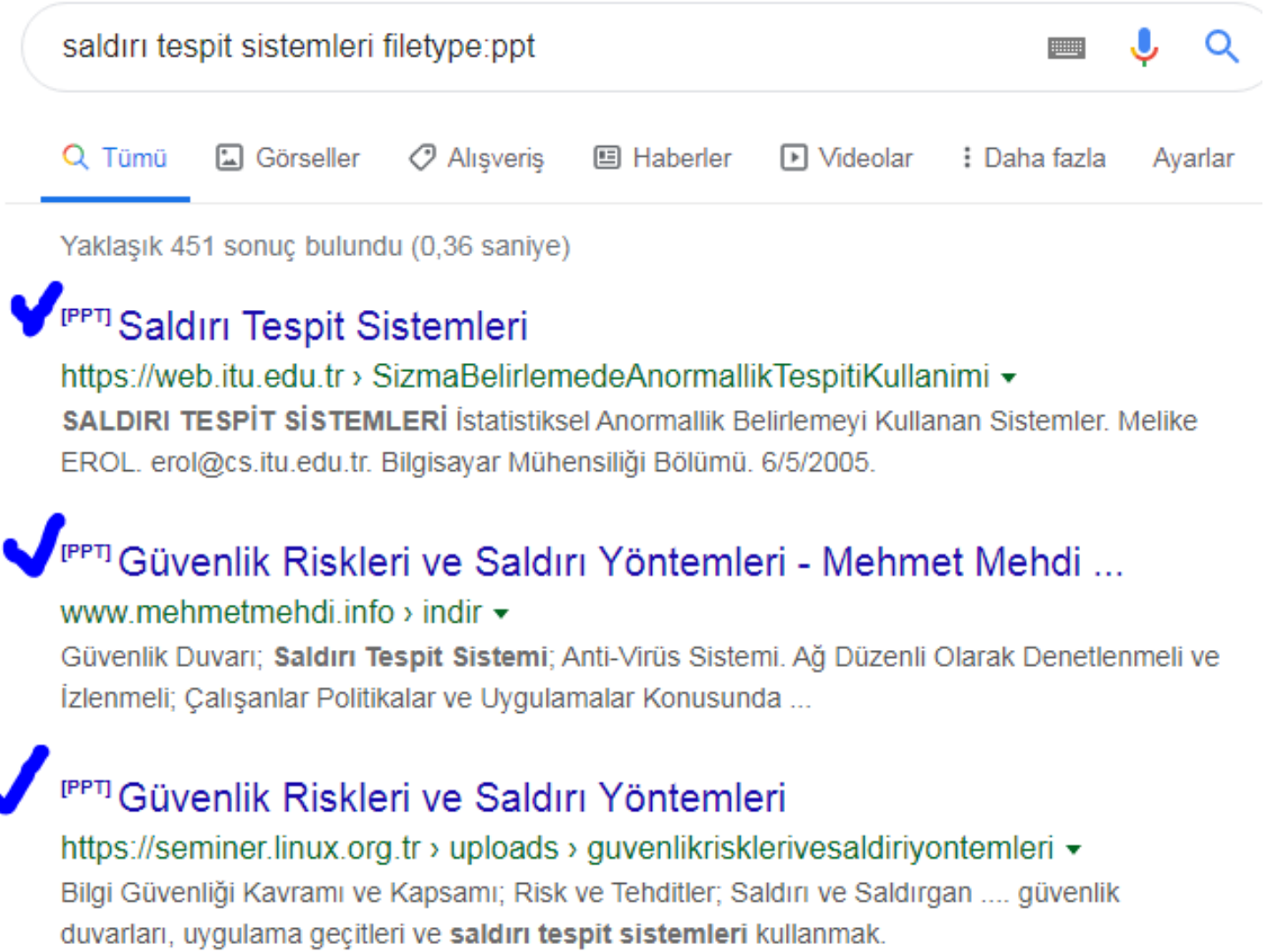


Google ile bazı arama teknikleri *filetype*:



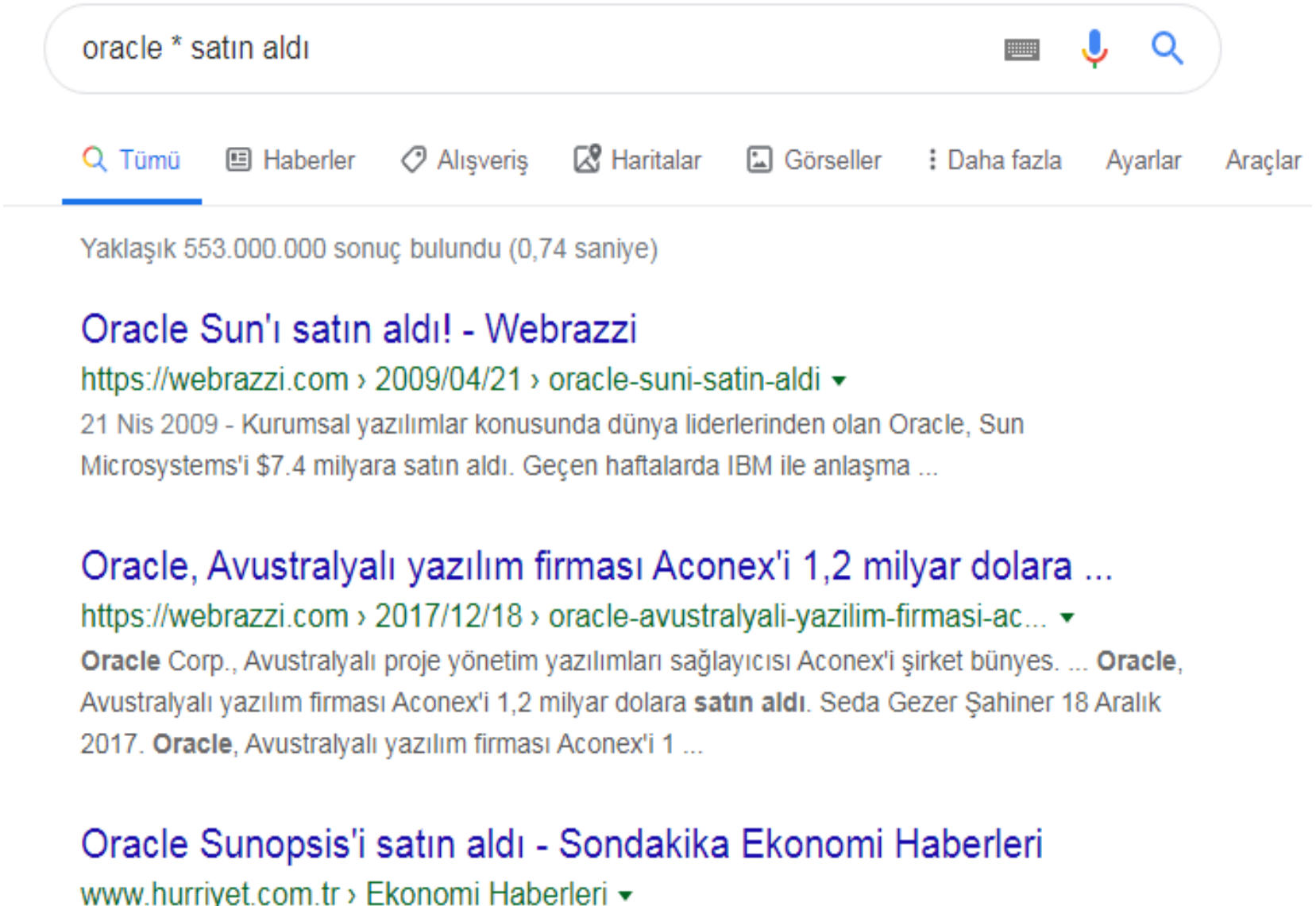
Google ile bazı arama teknikleri

filetype:



Google ile bazı arama teknikleri

joker *****



Proje Rapor Hazırlama
İntihal, benzerlik oranı, referans, kaynak…

Özet **– Bilişim dünyasındaki olağanüstü gelişmelere paralel olarak kişi, kurum ve kuruluşlar açısından bilgi sistemleri güvenliğinin öneminde büyük ölçüde artan bir süreç yaşanmaktadır. Bilgi sistemlerinde güvenliğin sağlanması için birçok farklı çalışmalar yapılmaktadır. Bu çalışmalar ile bilgi veya bilgisayar sistemlerinde güvenliğin sağlanması amaçlanmaktadır. Bu amaç doğrultusunda kullanılan güvenlik yazılımları kişi kurum ve kuruluşlara ait sistemlerin korunmasında büyük önem taşımaktadır. Bu makale çalışmasında, günümüzde yaygın olarak kullanılan güvenlik araçları detaylıca incelenmiş ve bu araçlar işlevleri, kullanım alanları gibi birçok özellikleri dikkate alınarak 22 farklı kategoride sunulmuştur. Bununla birlikte kişi, kurum ve kuruluşların bilgi sistemleri güvenliğinin sağlamasına yönelik çeşitli çözüm önerileri sıralanmıştır. Böylece bilgi sistemleri güvenliği konusunda önem arz eden sağlam güvenlik politikaları, yararlı güvenlik araçları ve çeşitli güvenlik önlemleri sunularak, kişisel veya kurumsal olarak ihtiyaç duyulan temel güvenlik stratejileri ortaya konulmuştur.**

Anahtar Kelimeler **– Bilgisayar Güvenliği, Bilgi Güvenliği Araçları, Bilgi Güvenliği, Casus Yazılımlar**

Bilgisayar ağ sistemlerinin yaygın olarak kullanılmaya başlanmasıyla birlikte ağ üzerinden yapılan saldırılarda da artışlar yaşanmaya başlanmıştır. Saldırı olaylarının bir kısmına bilinçsiz kullanıcılar neden olurken bir kısmına da bilerek sisteme zarar vermek isteyen kötü niyetli kişiler neden olmaktadır. 1980'li yıllarda bilgisayar haberleşmelerinde TCP/IP protokol ailesi dünya çapında kabul görmüş ve internet bu protokol aracılığı ile yaygınlaşmıştır. İnternetin yaygınlaşması ile bilgisayar haberleşmelerindeki atakların sayısı ve çeşidi de artmıştır [1]. Bu ataklar karşısında kimlik doğrulama, yetkilendirme, antivirüs programları gibi güvenlik çözümleri geliştirilmeye çalışılmıştır. İlk çıkan ataklar daha çok basit kod yürütme, parola tahminleri gibi etkisi ve olasılığı düşük ataklar olmasına karşın süreç içerisinde atakların karmaşıklığı ve etkileri artmıştır [2]. Buna karşın bu atakları kullanabilmek için gereken bilgi düzeyi düşmüştür. Çünkü bu bilgiler çoğunlukla internet üzerinden kontrolsüz olarak yayılmıştır. Şekil 1'de atakların zamana göre değişimleri, etkileri ve onları kullanabilmek için gerekli bilgi düzeyi görülmektedir [3].

Bilişim Etiği

1. Bilgisayar başka insanlara zarar vermek için kullanılamaz.
2. Başka insanların bilgisayar çalışmaları karıştırılamaz.
3. Bilgisayar ortamında başka insanların dosyaları karıştırılamaz.
4. Bilgisayar hırsızlık yapmak için kullanılamaz.
5. Bilgisayar yalan bilgiyi yaymak için kullanılamaz.
6. Bedeli ödenmeyen yazılım kopyalanamaz ve kullanılamaz.
7. Başka insanların bilgisayar kaynakları izin almadan kullanılamaz.
8. Başka insanların entelektüel bilgileri başkasına mal edilemez.
9. Kişi yazdığı programın sosyal hayata etkilerini dikkate almalıdır.
10. Kişi, bilgisayarı, diğer insanları dikkate alarak ve saygı göstererek kullanmalıdır.

İletişim Bilgilerim

- İletişim Bilgilerim
  - muhammetbaykara.com
  - mbaykara@firat.edu.tr,
  - 0 424 237 00 00-4296
  - https://www.linkedin.com/in/muhammetbaykara/
  - https://www.youtube.com/c/MuhammetBaykara



TEŞEKKÜRLER



YMT 312-Yazılım Tasarım ve Mimarisi

Yazılım Mühendisliği'ne Giriş

Bölüm-1

Fırat Üniversitesi Yazılım Mühendisliği Bölümü

Bu Haftaki Konular

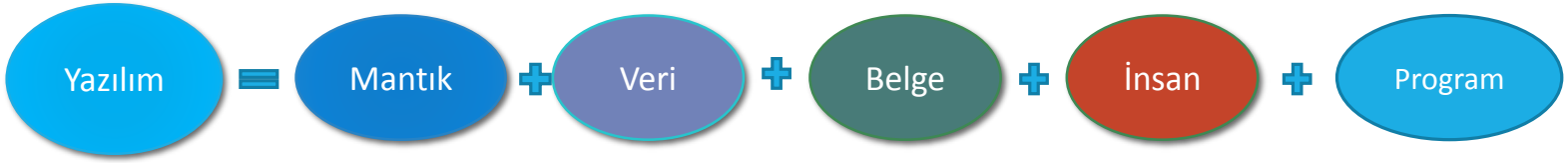
- Yazılım Nedir?.....4
- Yazılım Mühendisliği.....15
- Yazılımların Sınanması.....20
- Yazılım Maliyetleri.....22
- Yazılım Sistemlerinin Sınıflandırılması.....23
- Yazılımda Kalite.....27

Amaçlar

- Yazılım nedir?
- Yazılım ve Donanım karşılaştırılması?
- Yazılım Mühendisliği nedir?
- Yazılım Mühendisi kime denir?
- Yazılımda Hatalar?
- Yazılım Maliyetleri?
- Yazılımda Kalite?



Yazılım Nedir?



- Bileşenlerinin, belirli bir üretim amacına yönelik olarak bir araya getirilmesi, yönetilebilmesi için kullanılabilecek ve üretilen, yöntem, araç, bilgi ve belgelerin tümünü içerir.

- Yazılım en yalın biçimiyle, "**Bir sistemin donanım bileşenleri dışında kalan her şey**" olarak tanımlanabilir.

Mantık (Algoritma)

- Mantık, bilginin yapısını incelerken, kesin sonuca ulaşmak için doğru ile yanlış arasındaki akıl yürütme ayrımı yapmaktır.
- Bilgisayarlaştırılmak istenen işin mevcut mantığı yazılıma yansıtılmak durumundadır.
- Bu nedenle mantık (algoritma) bileşeni yazılımın en önemli bileşenlerinden biridir.



Veri (Bilgi)

- Veri (ing. data), işlenmemiş bilgi veya bilginin ham halidir. Bilgi ise, en basit anlamda verinin işlenmiş şeklidir. Her tür yazılım mutlaka bir veri üzerinde çalışmak durumundadır. Veri dış ortamdan alınabileceği gibi, yazılım içerisinde de üretilebilir.
- Yazılımın temel amacı “veri” yi “bilgi” ye 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.



İnsan (Kullanıcı, geliştirici)

- İki boyutludur; yazılımı geliştirenler ve kullananlar.
- Günümüzde artık tek kişi ile yazılım geliştirmekten söz edilmemektedir.
- Yazılım üretimi için bir takım oluşturulmakta ve takımın uyumlu çalışabilmesi için çeşitli yöntemler geliştirilmektedir.

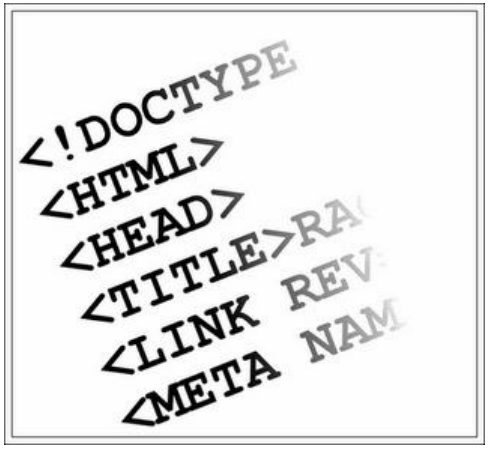


Program (Kod)

- Yazılımın ana çıktısı sonuçta bir bilgisayar programıdır.
- Program işletime alındıktan sonra bakım çalışmaları sürekli olarak gündeme gelir.

Bunun iki temel nedeni:

- Hiç bir program bütünüyle her olasılık göz önüne alınarak test edilemez.
- İşletmeler doğaları gereği dinamik bir yapıya sahiptir ve zaman içerisinde sürekli olarak yeni istek ve gereksinimler ortaya çıkabilmektedir.



Yazılım Donanım Karşılaştırması



- Yazılım geliştirilir & donanım üretilir. (fabrika ortamında seri üretim)
- Donanım bileşenleri dışarıdan temin edilebilir, ancak yazılımı oluşturan parçalar için bu çoğu zaman mümkün değildir (günümüzde "yeniden kullanılabilir yazılım" %1-2).



Yazılım Donanım Karşılaştırması

- Yazılım eskimez.

- Oysa, her donanımın belli bir ömrü vardır. Ömrünü tamamlayan donanım yenisi ile değiştirilir.

- Yazılımın eskimesi ortaya çıkabilecek yeni ihtiyaçları karşılayamaması, kullandığı teknolojinin eskimesi olarak tanımlanabilir.

- Yeni gereksinimler yazılıma ekler yaparak yansıtılır.

Yazılım Donanım Karşılaştırması

- Yazılım en az donanım kadar önemlidir.

- Diyaliz makinelerinde kullanılan yazılımların 2000 yılı uyumsuzluğundan ötürü, bir çok diyaliz makinesi çalışamamış ve böbrek hastaları zor durumda kalmıştır.

- Japonya'da telefon yazılımında ortaya çıkan bir yazılım hatası onbinlerce abonenin saatlerce telefon konuşması yapamamasına neden olmuştur.

Yazılım Donanım Karşılaştırması

- Yazılım kopyalama ve donanım kopyalama farklıdır.
- Hata toleransı amacıyla, hayati olan bir donanımın sistemde bir kopyası daha bulundurulur ve sistemde biri arızalandığında diğeri çalışmayı devralabilir.
- Oysa, bir yazılımı sistemde iki ayrı bilgisayar üzerine kopyalamak oluşabilecek hatalara çözüm olmayacaktır. Belki, sisteme aynı işi yapan iki farklı eş yazılım yüklenmesi çözüm olabilir (kritik yazılım sistemleri-uçak avionics).

Yazılım Üretim Ortamı

- Değişik yetenekte bir çok personel (analist, programcı, test uzmanı, vs.)
- Yazılım çıktısı ile ilgilenen kullanıcılar
- Yeniliğe tepki gösteren kullanıcılar ve yöneticiler
- Yeterince tanımlanmamış kullanıcı beklentileri
- Personel değişim oranının yüksekliği
- Yüksek eğitim maliyetleri
- Dışsal ve içsel kısıtlar (zaman, maliyet, işgücü, vs)
- Standart ve yöntem eksiklikleri
- Verimsiz kaynak kullanımı
- Mevcut yazılımlardaki kalitesizlik
- Yüksek üretim maliyeti



Yazılım Mühendisliği

IEEE Tanımı (1993)

“Yazılım Mühendisliği:

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

Diğer bir deyişle mühendisliğin yazılıma uygulanmasıdır.



Yazılım Mühendisliği

- Yazılım üretiminin mühendislik yöntemleriyle yapılmasını öngören ve bu yönde;
  - yöntem,
  - araç
  - teknik ve
  - metodolojiler

  üreten bir disiplindir.



Yazılım Mühendisliği

- Yazılım mühendisliği bir yöntemler, teknikler ve araçlar kümesi olarak değerlendirilebilir.
- Yazılım mühendisliğinin hedefi; yazılım üretimindeki karmaşıklıkları gidermektir.
- Geçmişte kullanılan iş akış şemaları gibi yöntemler günümüzde yetersiz kalmaktadır.
- Ayrıca, yazılım üretimi işi tek kişinin başarabileceği boyuttan çıkmış ve bir takım işi biçimine dönüşmüştür.

Yazılım Mühendisi

- Yazılım Mühendisliği İşini yapan kişidir.

- Temel hedefi; üretimin en az maliyet ve en yüksek nitelikte yapılmasını sağlamaktır.

- Programcı değildir. Ancak programcının tüm yeteneklerine sahiptir.

- Yazılımın daha çok mantıksal boyutuyla ilgilenir ve işi insanlarla ilişkiyi gerektirir.

- Sistem analisti de değildir. Farkı; analist sadece sistemin analiz aşaması ile ilgilenirken, yazılım mühendisi tüm aşamaların içindedir.

Yazılım Hataları

- Yazılım hataları, yazılım yaşam döngüsünde çok önemli yer tutan unsurlardan biridir.
- Yazılım geliştirme de karşılaşılan en sıkıcı durumdur.
- Hatalar yüzünden yazılım maliyeti artmaktadır.
- Yazılım geliştirme sürecini uzatmaktadır.
- Çözümü erken bulunmayan hatalar bazen uyulması gereken sistemi zor durumda bırakarak zamansal problemler oluşturmaktadır



Yazılımların Sınanması

- Bir programı tüm ayrıntıları ile test etmek teorik olarak mümkün olmakla birlikte, uygulamada bu mümkün değildir.
- Yazılım ancak sınırlı sayıda veri ile sınanabilir.

Mantıksal Tasarım	%20
İşlevsel Tasarım	%15
Kodlama	%30

Hataların “Yayılma” Özelliği

- Yazılımda Hata Düzeltme Maliyetleri:
  - Yazılım üretimindeki hatalar yayılma özelliği gösterir.
  - Bu nedenle, hata düzeltme maliyetleri ilerleyen aşamalarda giderek artar.

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



Yazılım Maliyetleri

- Günümüzde yazılım maliyetlerindeki artışlar giderek artmaktadır. "Yazılımımızı alırsanız yanında donanımı ücretsiz olarak sağlayacağız" deyişi zaman içerisinde giderek doğrulanmaktadır.
- Örneğin, günümüzde bir kopyası yüz bin dolar dolayında satılan kurumsal kaynak planlama yazılımlarının bulunduğu gözlemlenmektedir.
- Öte yandan bir kişisel bilgisayar ise 1000 ABD dolarının altında satılmaktadır.
- Yazılımın kopyalanma maliyeti ile donanım kopyalama maliyetinin arasındaki farklılık dikkate alındığında, yazılım maliyetlerinin, donanım maliyetlerine oranla oldukça yüksek olduğu ortaya çıkar.



Yazılım Sistemlerin Sınıflandırılması

- **İşlevlerine göre sınıflandırma**
- **Zamana dayalı özelliklere göre sınıflandırma**
- **Boyuta göre sınıflandırma**



İşlevlerine Göre Sınıflandırma

Hesaplama	Mühendislik Çözümleme
Veri İşleme	Bankacılık
Süreç Temelli	Gömülü Sistemler
Kural Temelli	Robotik, Yapay Zekâ
CAD	Sinyal İşleme

Zamana Dayalı Özelliklere Göre Sınıflandırma

Toplu (Çevrim-Dışı)	Çevrim-İçi
Gerçek Zamanlı	

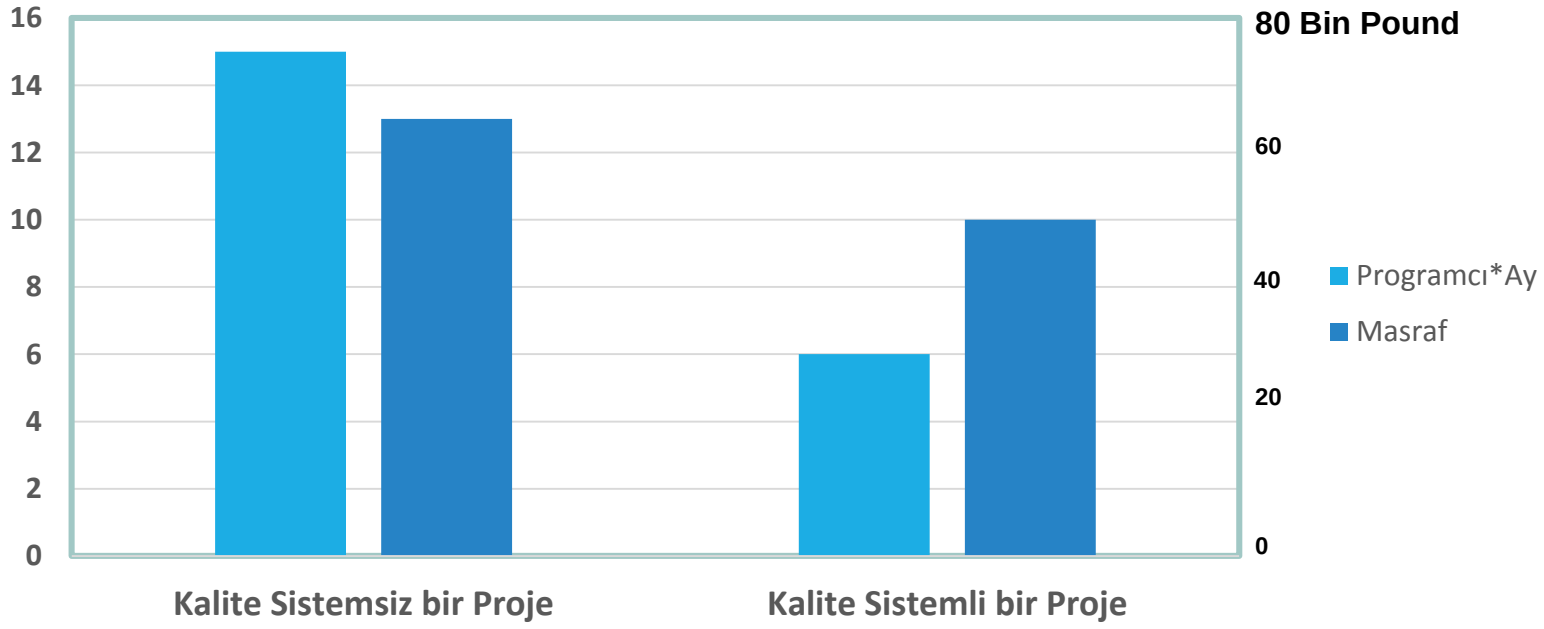




Boyuta Göre Sınıflandırma

Küçük (SS<2000)	PC Oyunları Öğrenci Projeleri
Orta (2000<SS<100,000)	CAD BDE Yazılımları
Büyük(100,000<SS<1 Milyon)	İşletim Sistemleri
Çok Büyük (SS>1 Milyon)	Komuta Kontrol Sistemleri Hava Tahmini Sistemleri Yıldız Savaşları Sistemleri

Yazılımda Kalite



Yazılımda Kalite

- Üretim Süreci Boyunca ara ürünlere ilişkin kalite standartlarının geliştirilmesi ve geliştirme işlemlerinin bu standartlara uygunluğunun denetlenmesidir.

Yazılım kalite sağlama etkinlikleriyle;

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



Yazılımda Kalite

Ekonomi	Tamlık	Yeniden Kullanılabilirlik	Etkinlik	Bütünlük
Güvenirlik	Modülerlik	Belgeleme	Kullanılabilirlik	Temizlik
Değiştirilebilirlik	Geçerlik	Esneklik	Genellik	Sınanabilirlik
Taşınabilirlik	Bakılabilirlik	Anlaşılabilirlik	Birlikte Çalışabilirlik	

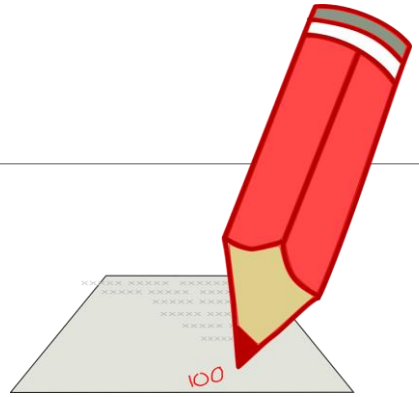
Özet

- **Yazılım**= Mantık, veri, belge, insan ve program bileşenlerinin, belirli bir üretim amacına yönelik olarak bir araya getirilmesi, yönetilebilmesi için kullanılabilecek ve üretilen, yöntem, araç, bilgi ve belgelerin tümünü içerir.
- **Yazılım Mühendisliği**: Sistemli, düzenli, ölçülebilir bir yaklaşımın yazılım geliştirmede, yazılımın işlenilmesinde ve bakımında uygulanmasıdır. Diğer bir deyişle mühendisliğin yazılıma uygulanmasıdır.
- **Yazılım mühendisliğinin hedefi**; yazılım üretimindeki karmaşıklıkları gidermektir.
- **Yazılım kalite sağlama etkinlikleriyle;**
  - Yazılım maliyetleri düşürülür,
  - Yazılım üretiminin yönetimi kolaylaşır,
  - Belgeleme ve standart sorunları giderilir.

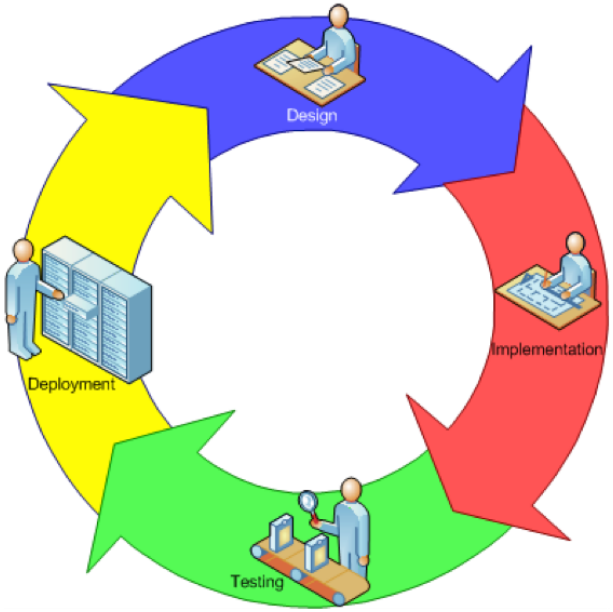
Sorular

1. Yazılım ile program arasındaki farklılığı belirtiniz.
2. Yazılım ile donanım arasındaki farklılıkları belirtiniz.
3. Neden yazılım giderek pahalılaşırken, donanım ucuzlamaktadır?
4. Yazılım mühendisliği ile diğer mühendislik disiplinlerini karşılaştırınız.
5. Bu bölümde verilen yazılım sınıflandırmasını dikkate alarak, her sınıflandırma için bir yazılım örneği veriniz.
6. Yazılım mühendisi ile programcı arasındaki farklılığı belirtiniz.
7. Yazılım Hatalarında Yayılma neyi ifade etmektedir?
8. Yazılım Maliyetinde hangi Sınıflandırma ağır basmaktadır?
9. İşlevlerine Göre Sınıflandırma da Kural Temelli ’ye Örnek veriniz.
10. Yazılım Kalitesini Tanımlayınız.

Ödev



1. Yazılım Yaşam Döngüsünü Araştırınız.
2. Yazılım Süreç Modellerini Araştırınız.



Kaynaklar

- "Software Engineering A Practitioner's Approach" (7th. Ed.), Roger S. Pressman, 2013.
- "Software Engineering" (8th. Ed.), Ian Sommerville, 2007.
- *"Guide to the Software Engineering Body of Knowledge", 2004.*
- " Yazılım Mühendisliğine Giriş", TBİL-211, Dr. Ali Arifoğlu.
- "Yazılım Mühendisliği" (2. Basım), Dr. M. Erhan Sarıdoğan, 2008, İstanbul: Papatya Yayıncılık.
- Kalıpsiz, O., Buharalı, A., Biricik, G. (2005). Bilgisayar Bilimlerinde Sistem Analizi ve Tasarımı Nesneye Yönelik Modelleme. İstanbul: Papatya Yayıncılık.
- Buzluca, F. (2010) Yazılım Modelleme ve Tasarımı ders notları (http://www.buzluca.info/dersler.html)
- Hacettepe Üniversitesi BBS-651, A. Tarhan, 2010.
- Yazılım Proje Yönetimi, Yrd. Doç. Dr. Hacer KARACAN

http://blog.alisuleymantopuz.com/2014/08/30/yazilim-mimarisi-ve-tasarimi-nedir/ * http://www.akifsahman.com/?p=175

https://ece.uwaterloo.ca/~se464/08ST/index.php?src=lecture * http://info.psu.edu.sa/psu/cis/azarrad/se505.htm

http://www.metinakbulut.com/YAZILIM-MIMARISI/ * http://ceng.gazi.edu.tr/~hkaracan/source/YPY_H3.pdf

http://iiscs.wssu.edu/drupal/node/3399 * http://www.cs.toronto.edu/~sme/CSC340F/slides/21-architecture.pdf

http://www.users.abo.fi/lpetre/SA10/ *http://sulc3.com/model.html

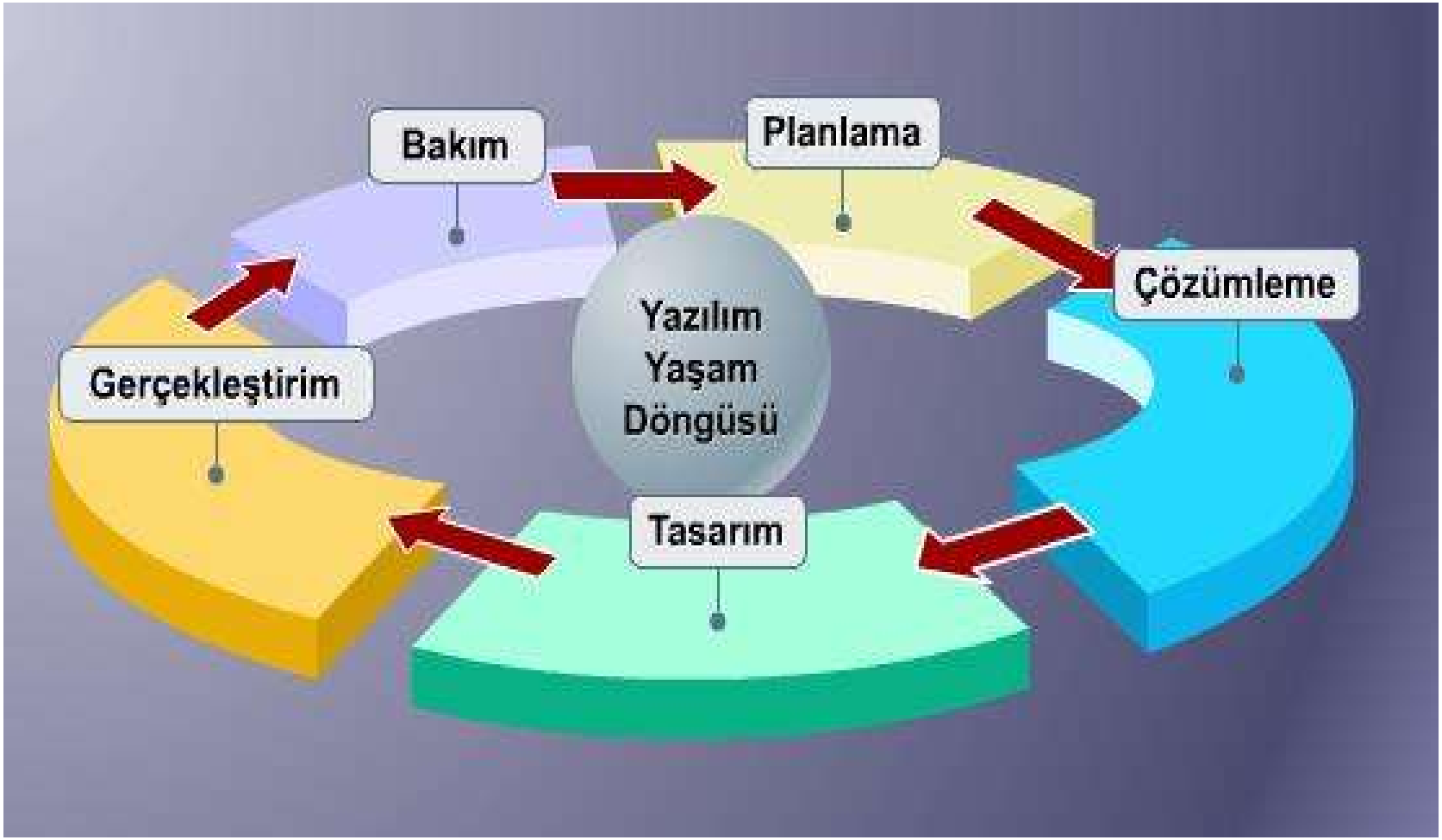
http://salyangoz.com.tr/blog/2013/11/23/digerleri/yazilim-gelistirme-surec-modelleri-3/

Hedef

- Bu bölümde, yazılım geliştirme yaşam döngüsünün çekirdek aşamaları olan, **planlama**, **çözümleme**, **tasarım**, **gerçekleştirim** ve **bakım** aşamaları tanıtılmakta, yaşam döngüsü modelleri ya da süreç modelleri olarak da **yaşam döngüsü**nün (life cycle) uygulamada kullanım modellerinin açıklanması hedeflenmektedir.
- Metododoloji, kavramı bu bölümde açıklanmaktadır.

Yazılım Yaşam Döngüsü

- Planlama, çözümleme, tasarım, gerçekleştirim, bakım



Yazılım Yaşam Döngüsü

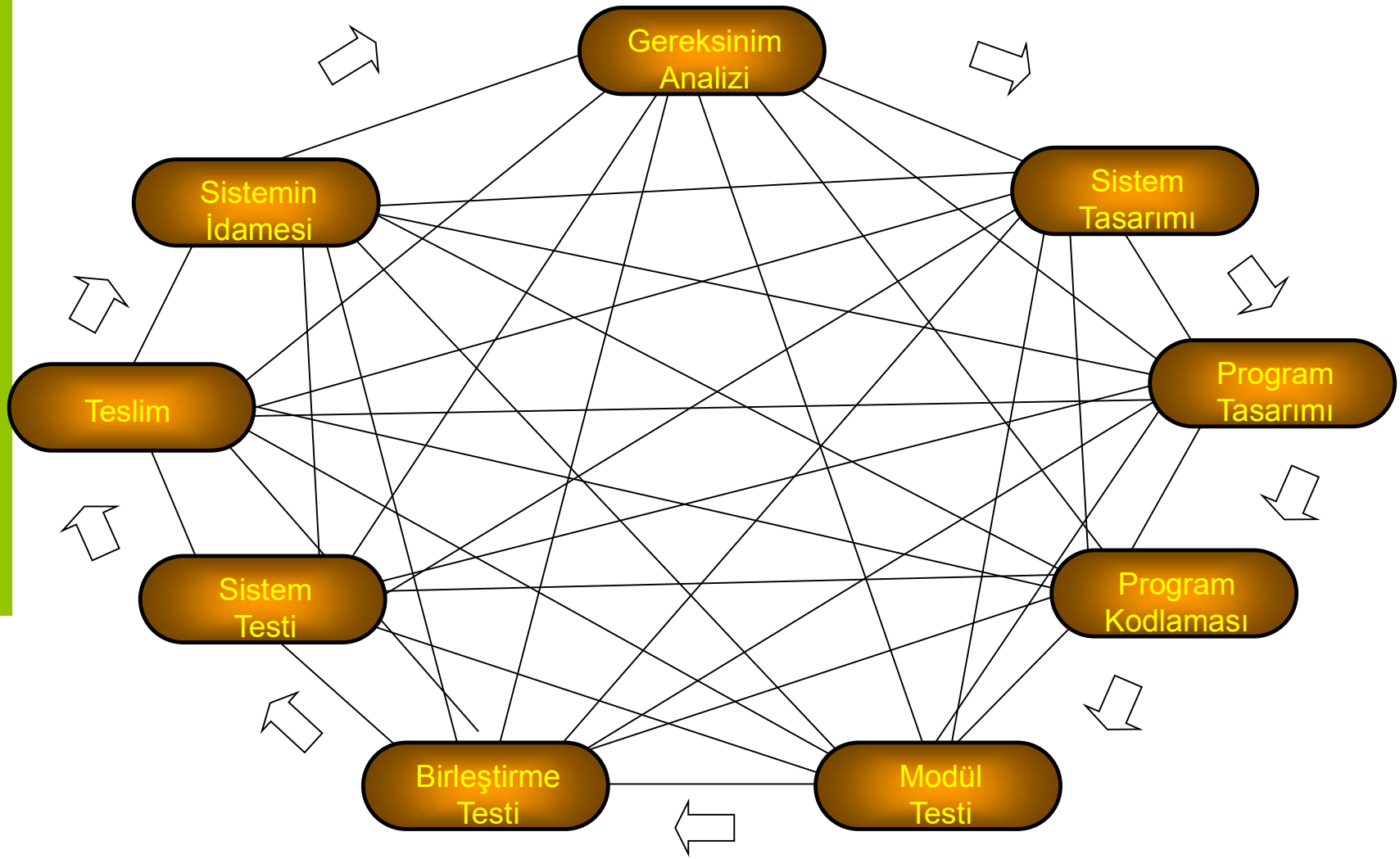
- Herhangi bir yazılımın, **üretim aşaması** ve **kullanım aşaması** birlikte olmak üzere geçirdiği tüm aşamalar biçiminde tanımlanır.
- Yazılım işlevleri ile ilgili gereksinimler sürekli olarak değiştiği ve genişlediği için, söz konusu aşamalar bir döngü biçiminde ele alınır.
- Döngü içerisinde herhangi bir aşamada geriye dönmek ve tekrar ilerlemek söz konusudur.
- Bu nedenle yazılım yaşam döngüsünün tek yönlü ve doğrusal olduğu düşünülmemelidir.

Yazılım Yaşam Döngüsü-Ne sağlar?

- Mühendislik faaliyetlerinin (üretim, işletme ve bakım aşamaları) yazılıma uyarlanması
- Üretimden kullanıma kadar tüm sürecin fazlara ayrılması ve böylece yönetim kolaylığı
- Her bir aşamada ne yapılacak?
- Her bir aşamadaki standartlara uyumluluk ile kaliteli yazılım geliştirme vb.



Gerçek Hayatta Program Geliştirme



Yazılım Yaşam Döngüsü Temel Adımları

Planlama: Personel ve donanım gereksinimlerinin çıkarıldığı, fizibilite (olurluk-yapılabilirlik) çalışmasının yapıldığı ve proje planının oluşturulduğu aşamadır.

Analiz: Sistem gereksinimlerinin ve işlevlerinin ayrıntılı olarak çıkarıldığı aşama. Mevcut var olan işler incelenir, temel sorunlar ortaya çıkarılır.

Tasarım: Belirlenen gereksinimlere yanıt verecek yazılım sisteminin temel yapısının oluşturulduğu aşamadır.

mantıksal; önerilen sistemin yapısı anlatılır,olası örgütsel değişiklikler önerilir.

fiziksel; yazılımı içeren bileşenler ve bunların ayrıntıları.

Gerçekleştirim: Kodlama, test etme ve kurulum çalışmalarının yapıldığı aşamadır.

Bakım: Hata giderme ve yeni eklentiler yapma aşaması (teslimden sonra).

Yazılım Yaşam Döngüsü Temel Adımları

- Yaşam döngüsünün temel adımları çekirdek süreçler (core processes) olarak da adlandırılır.

- Bu süreçlerin 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

- Süreç Akışı İçin Kullanılan Belirtim Yöntemleri

  Süreçler arası ilişkilerin ve iletişimin gösterildiği yöntemler (Veri Akış Şemaları, Yapısal Şemalar, Nesne/Sınıf Şemaları).

- Süreç Tanımlama Yöntemleri

  Süreçlerin iç işleyişini göstermek için kullanılan yöntemler (Düz Metin, Algoritma, Karar Tabloları, Karar Ağaçları, Anlatım Dili).

- Veri Tanımlama Yöntemleri

  Süreçler tarafından kullanılan verilerin tanımlanması için kullanılan yöntemler (Nesne İlişki Modeli, Veri Tabanı Tabloları, Veri Sözlüğü).

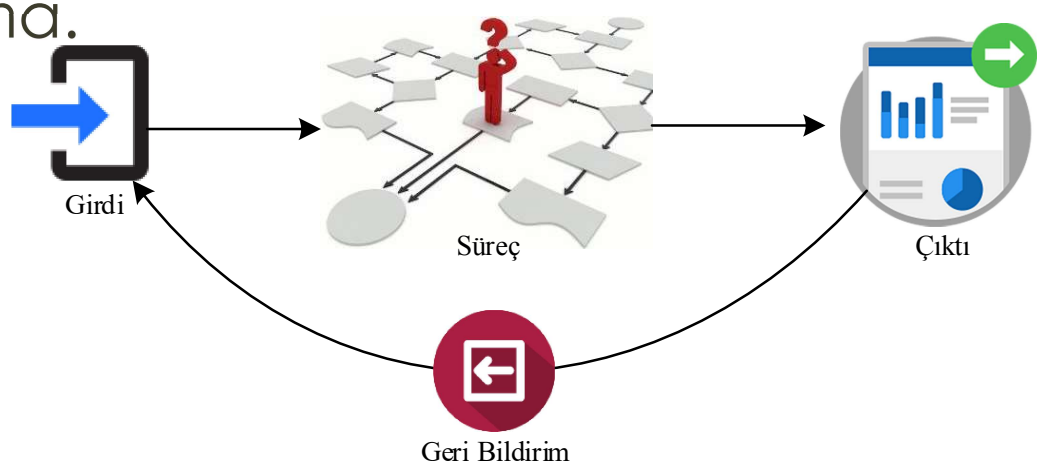
Yazılım Süreci Modelleri

- Yazılım üretim işinin genel yapılma düzenine ilişkin rehberlerdir.
- Süreçlere ilişkin ayrıntılarla ya da süreçler arası ilişkilerle ilgilenmezler.

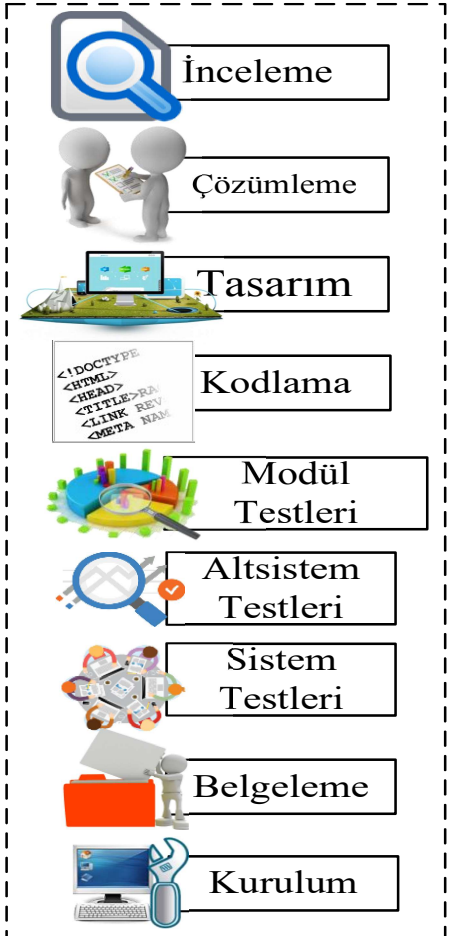
1. Gelişigüzel Model
2. Barok Modeli
3. Çağlayan (Şelale) Modeli
4. V Modeli
5. Helezonik (Spiral) Model
6. Evrimsel Model
7. Artırımsal Model
8. Araştırma Tabanlı Model

Gelişigüzel Model

- Herhangi bir model ya da yöntem yok.
- Geliştiren kişiye bağımlı (belli bir süre sonra o kişi bile sistemi anlayamaz hale gelebilir ve geliştirme güçlüğü yaşar).
- İzlenebilirliği ve bakımı oldukça zor.
- 60'lı yıllarda kullanılan ilkel model
- Genellikle tek kişilik üretim ortamı.
- Basit programlama.



Barok Modeli



Yaşam döngüsü temel adımlarının doğrusal bir şekilde geliştirildiği model.

70'li yıllar.

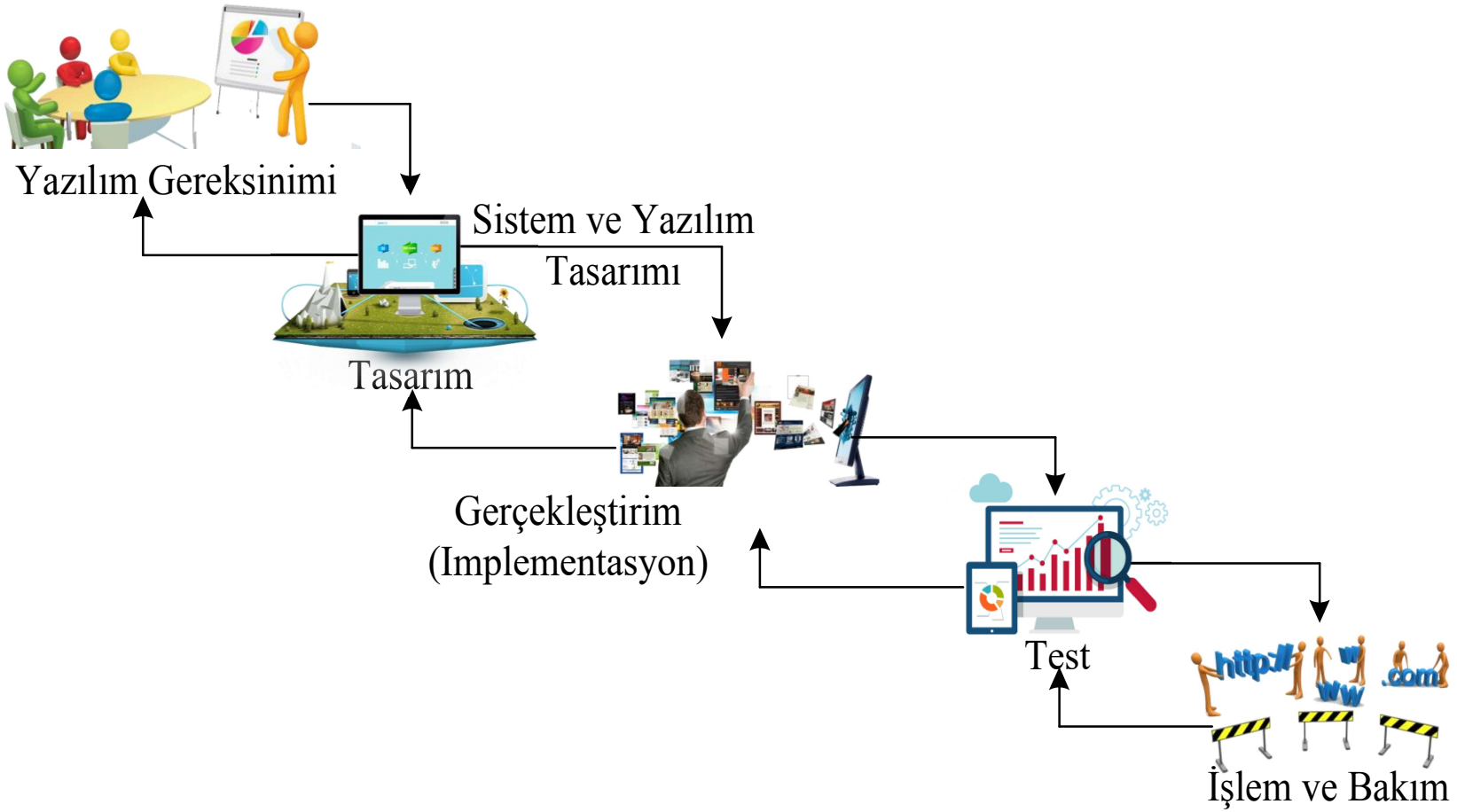
Belgelemeyi ayrı bir süreç olarak ele alır, ve yazılımın geliştirilmesi ve testinden hemen sonra yapılmasını öngörür.

Halbuki, günümüzde belgeleme yapılan işin doğal bir ürünü olarak görülmektedir.

Aşamalar arası geri dönüşlerin nasıl yapılacağı tanımlı değil.

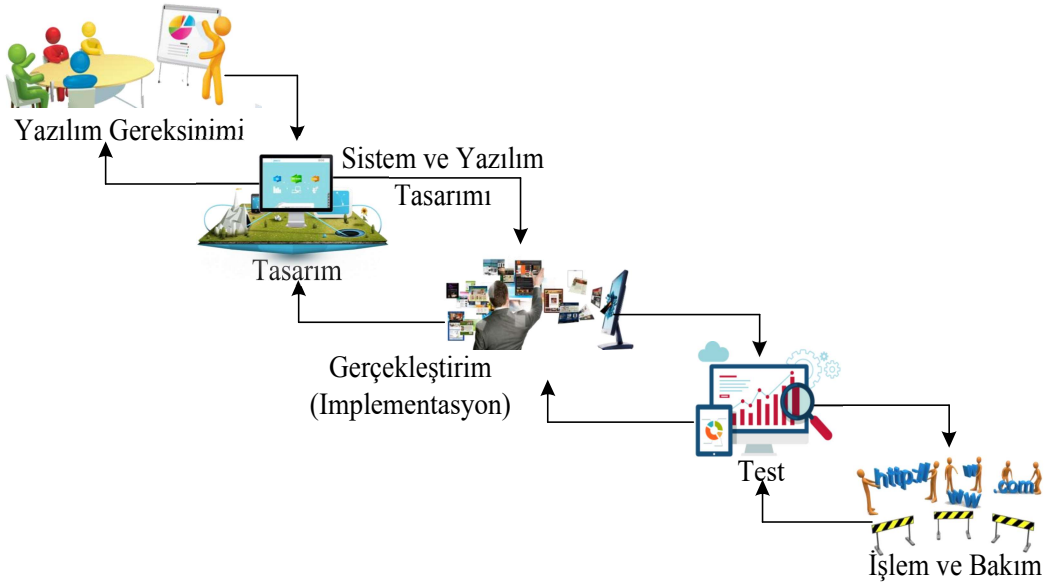
Gerçekleştirim aşamasına daha fazla ağırlık veren bir model olup, günümüzde kullanımı önerilmemektedir

Çağlayan (Şelale) Modeli



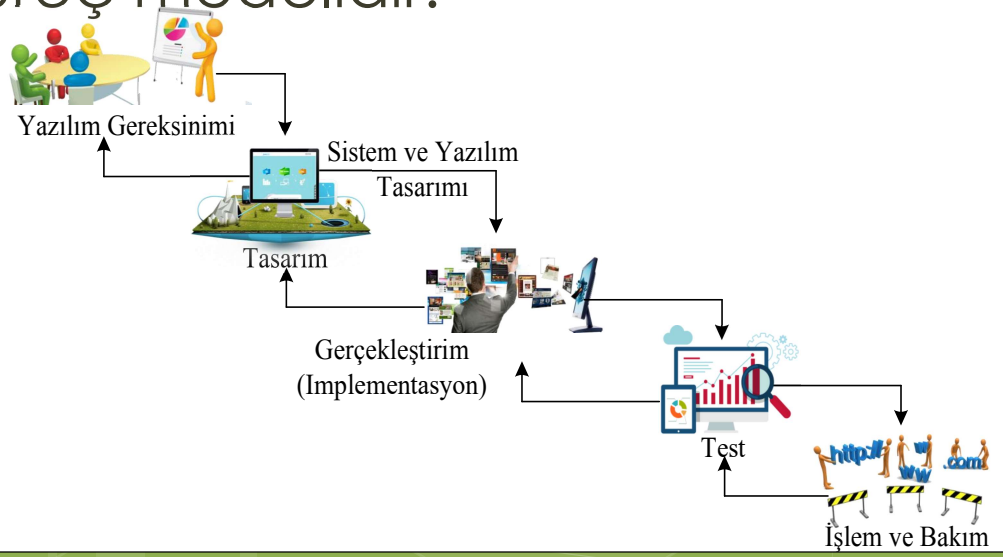
Çağlayan (Şelale) Modeli

- Yaşam döngüsü temel adımları baştan sona en az bir kez izleyerek gerçekleştirilir.
- İyi tanımlı projeler ve üretimi az zaman gerektiren yazılım projeleri için uygun bir modeldir.
- Geleneksel model olarak da bilinen bu modelin kullanımı günümüzde giderek azalmaktadır.



Çağlayan (Şelale) Modeli

- Barok modelin aksine belgeleme işlevini ayrı bir aşama olarak ele almaz ve üretimin doğal bir parçası olarak görür.
- Barok modele göre geri dönüşler iyi tanımlanmıştır.
- Yazılım tanımlamada belirsizlik yok (ya da az) ise ve yazılım üretimi çok zaman almayacak ise uygun bir süreç modelidir.



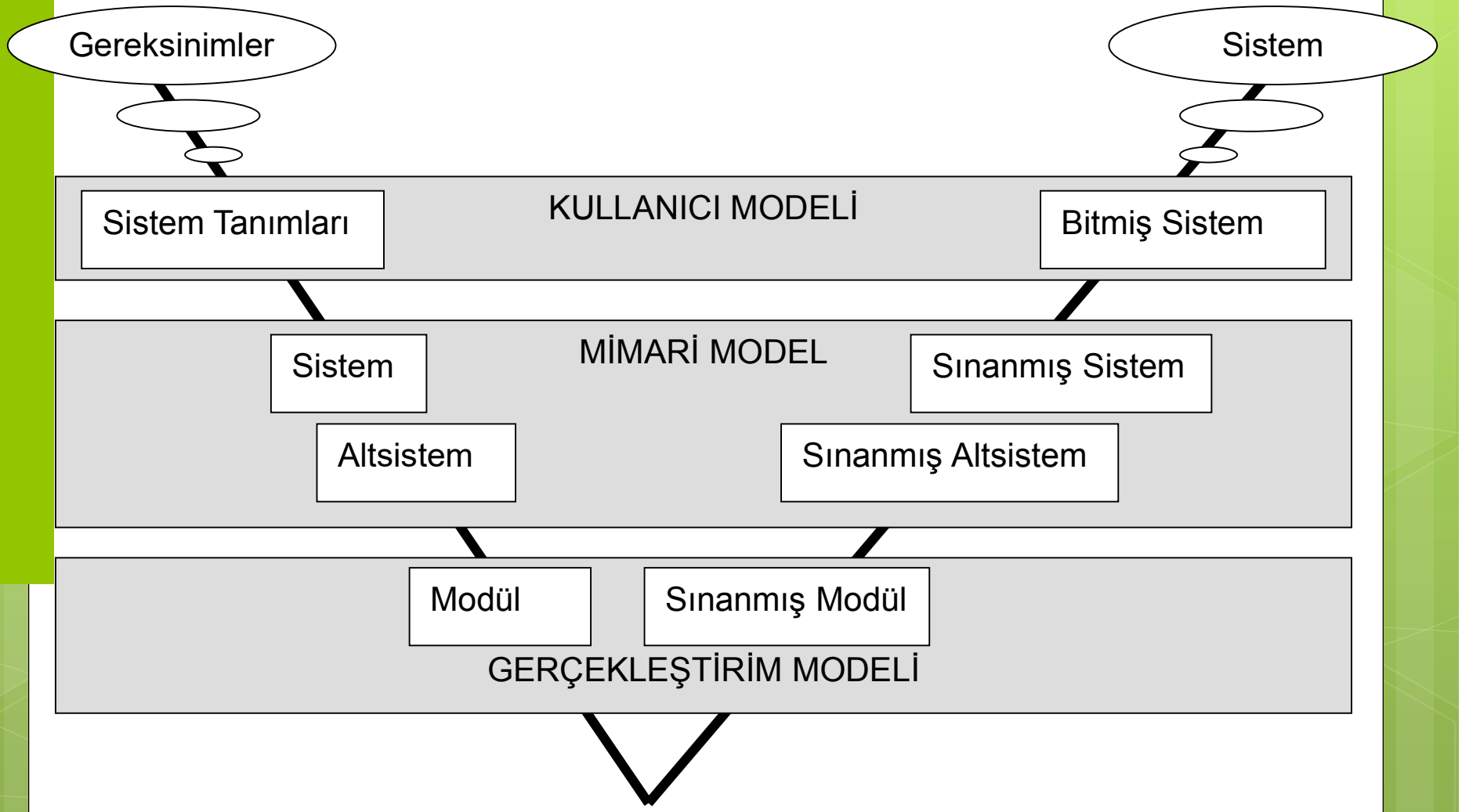
Avantajları

- Müşteriler ve son kullanıcılar tarafından da iyi bilinen anlaşılabilen adımlardan oluşur.
- İterasyonlar (tekrarlamalar) bir sonraki ve bir önceki adımlarla gerçekleşir, daha uzak adımlarla olması nadirdir.
- Değişiklik süreci yönetilebilir birimlere bölünmüştür.
- Gereksinim adımı tamamlandıktan sonra sağlam bir temel oluşur. Aşamaları iyi anlaşılabilir.
- Proje yöneticileri için işin dağılımını yapma açısından kolaydır.
- Gereksinimleri iyi anlaşılabilen projelerde iyi çalışır.
- Kalite gereksinimlerinin bütçe ve zaman kısıtlamasına göre çok daha önemli olduğu projelerde iyi çalışır.

Sorunları

- Gerçek yaşamdaki projeler genelde yineleme gerektirir.
- Genelde yazılımın kullanıcıya ulaşma zamanı uzundur.
- Gereksinim tanımlamaları çoğu kez net bir şekilde yapılamadığından dolayı, yanlışların düzeltilme ve eksiklerin giderilme maliyetleri yüksektir.
- Yazılım üretim ekipleri bir an önce program yazma, çalıştırma ve sonucu görme eğiliminde olduklarından, bu model ile yapılan üretimlerde ekip mutsuzlaşmakta ve kod yazma dışında kalan (ve iş yükünün %80'ini içeren) kesime önem vermemektedirler.
- Üst düzey yönetimlerin ürünü görme süresinin uzun oluşu, projenin bitmeyeceği ve sürekli gider merkezi haline geldiği düşüncesini yaygınlaştırmaktadır.

V Süreç Modeli



V Süreç Modeli

Sol taraf üretim, sağ taraf sınama işlemleridir.

V süreç modelinin temel çıktıları;

- Kullanıcı Modeli

  Geliştirme sürecinin kullanıcı ile olan ilişkileri tanımlanmakta ve 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.

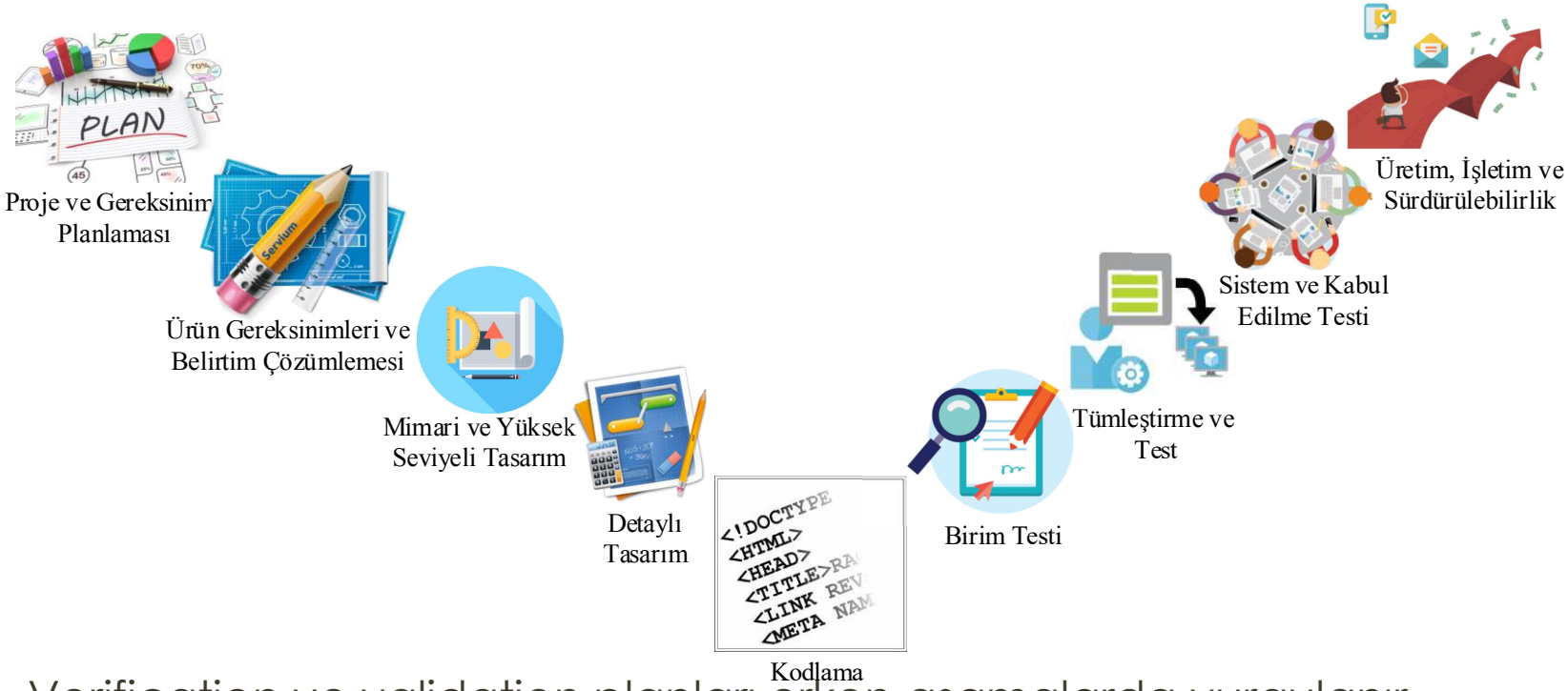
- Gerçekleştirim Modeli

  Yazılım modüllerinin kodlanması ve sınanmasına ilişkin fonksiyonlar.

V Süreç Modeli

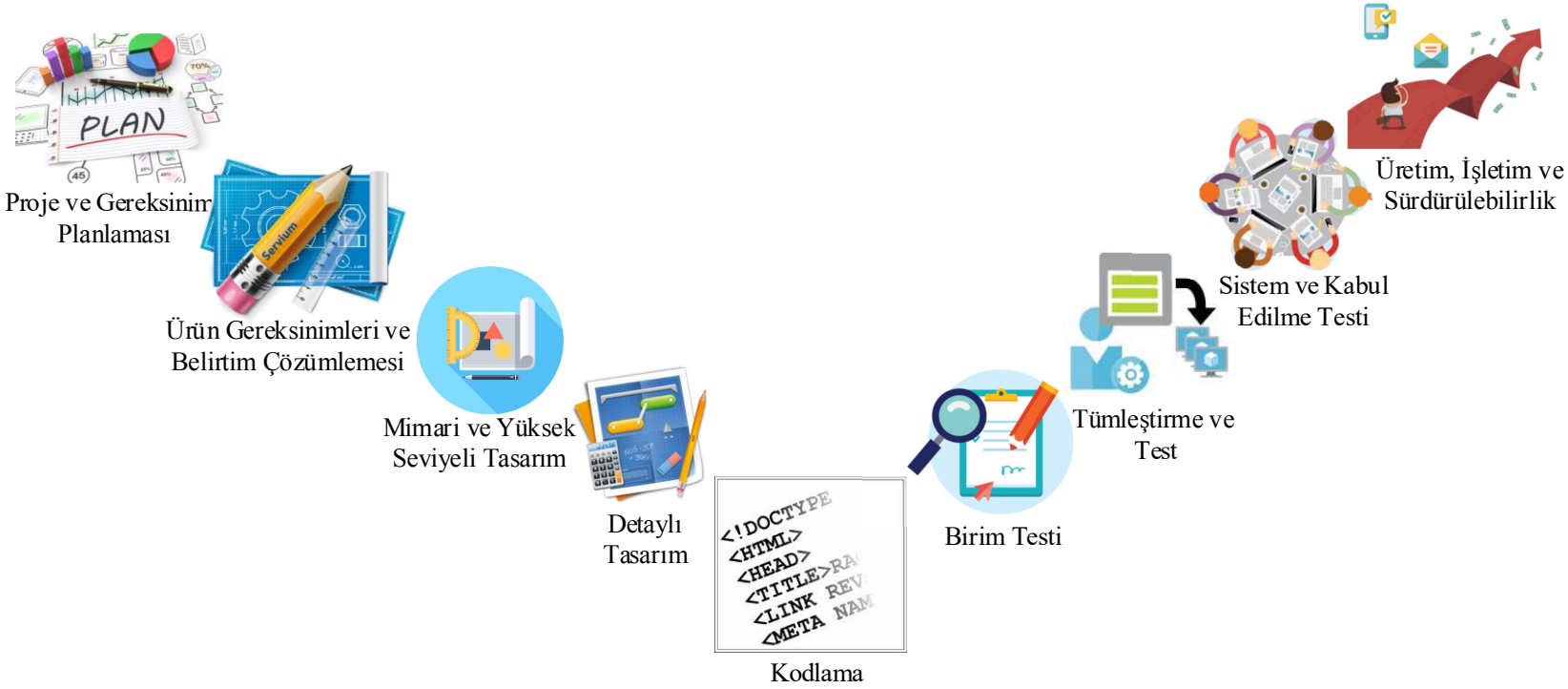
- Belirsizliklerin az, iş tanımlarının belirgin olduğu BT projeleri için uygun bir modeldir.
- Model, kullanıcının projeye katkısını arttırmaktadır.
- BT projesinin iki aşamalı olarak ihale edilmesi için oldukça uygundur:
  - İlk ihalede kullanıcı modeli hedeflenerek, iş analizi ve kabul sınamalarının tanımları yapılmakta,
  - İkinci ihalede ise ilkinde elde edilmiş olan kullanıcı modeli tasarlanıp, gerçeklenmektedir.

V Modeli-Avantajlar



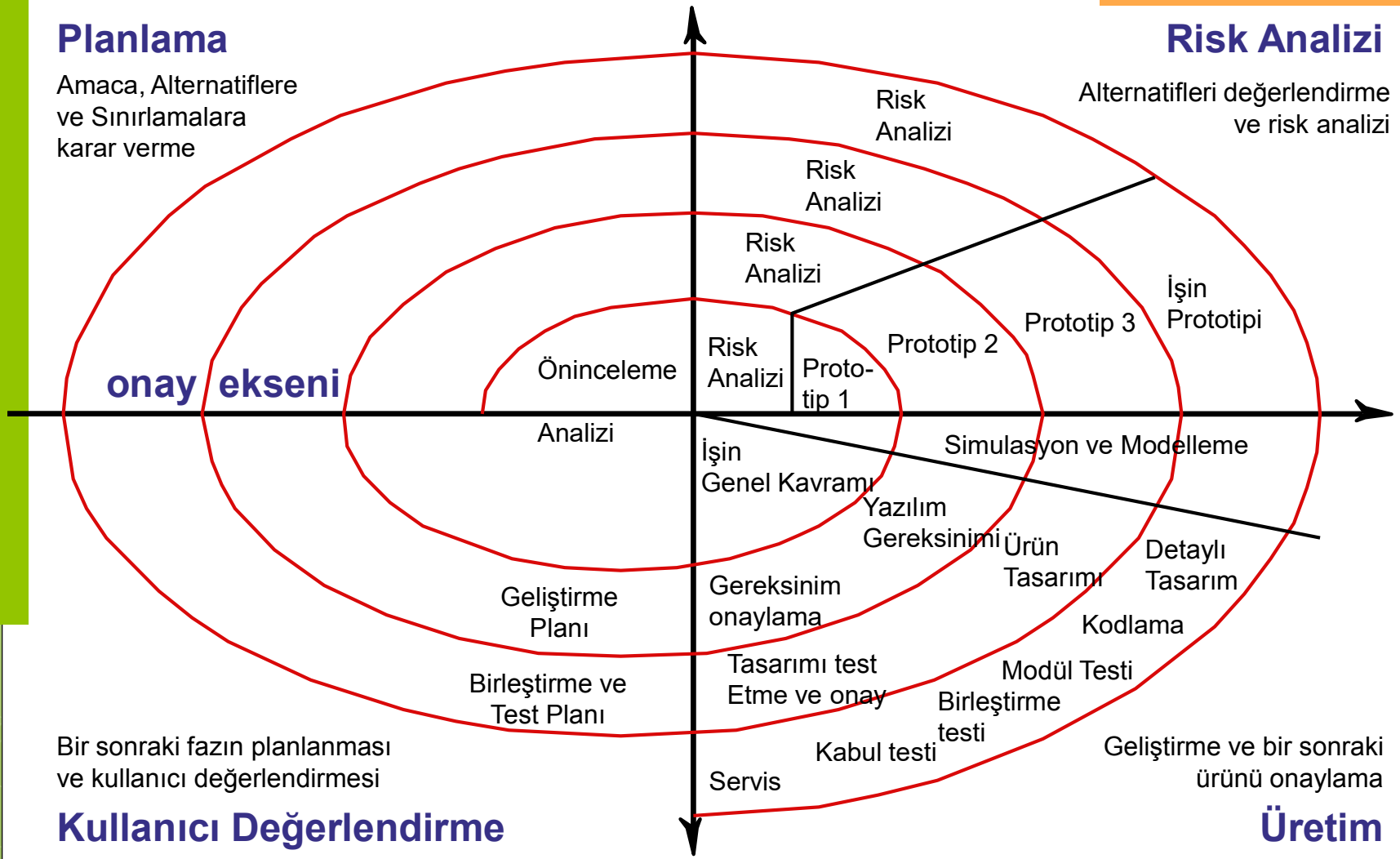
- Verification ve validation planları erken aşamalarda vurgulanır.
- Verification ve validation sadece son üründe değil tüm teslim edilebilir ürünlerde uygulanır.
- Proje yönetimi tarafında takibi kolaydır
- Kullanımı kolaydır.

V Modeli-Dezavantajlar



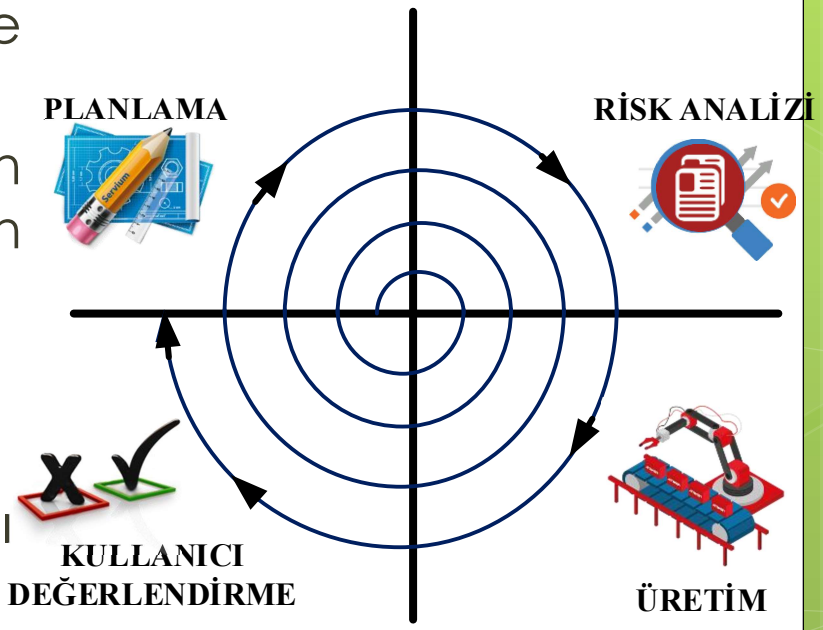
- Aynı zamanda gerçekleştirilebilecek olaylara kolay imkan tanımaz.
- Aşamalar arasında tekrarlamaları kullanmaz.
- Risk çözümleme ile ilgili aktiviteleri içermez.

Helezonik(Spiral) Modeli



Helezonik Model

1. **Planlama:** Üretilecek ara ürün için planlama, amaç belirleme, bir önceki adımda üretilen ara ürün ile 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 modelin avantajları

1. Kullanıcı Katkısı

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

2. Yönetici Bakışı

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

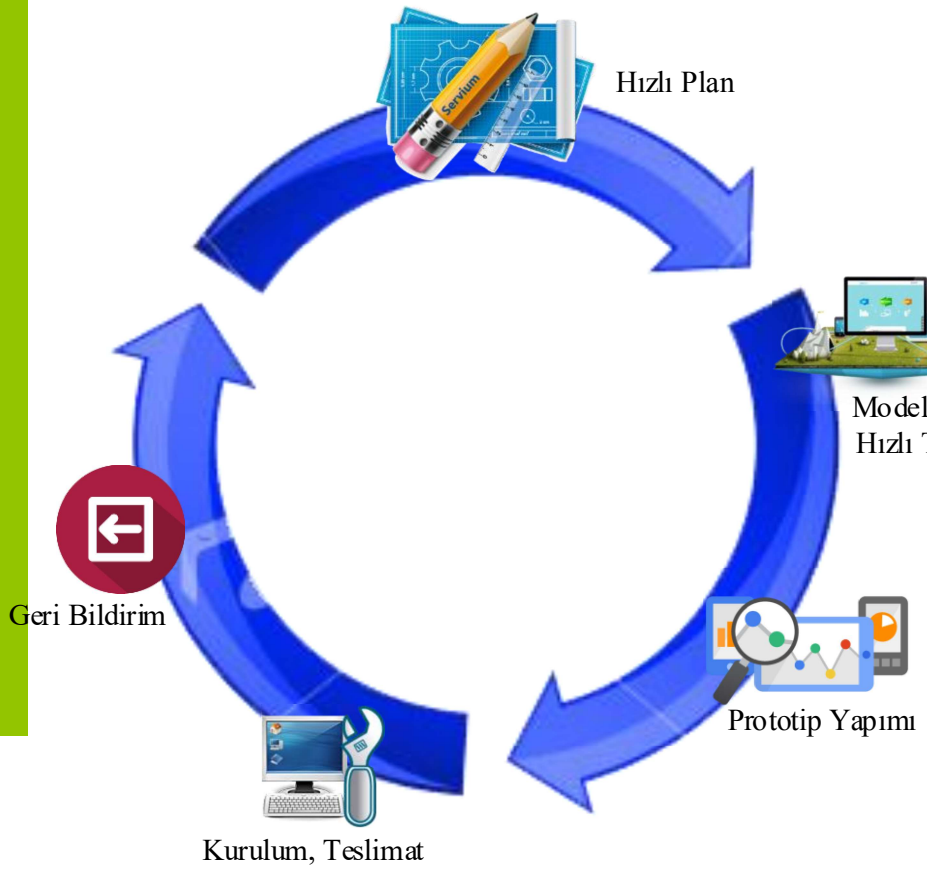
3. Yazılım Geliştirici (Mühendis) Bakışı

   Yazılımın kodlanması ve sınanması daha erken başlar.

Helezonik Model

- Risk Analizi Olgusu ön plana çıkmıştır.
- Her döngü bir fazı ifade eder. Doğrudan tanımlama, tasarım,... vs gibi bir faz yoktur.
- Yinelemeli artımsal bir yaklaşım vardır.
- Prototip yaklaşımı vardır.

Prototipleme



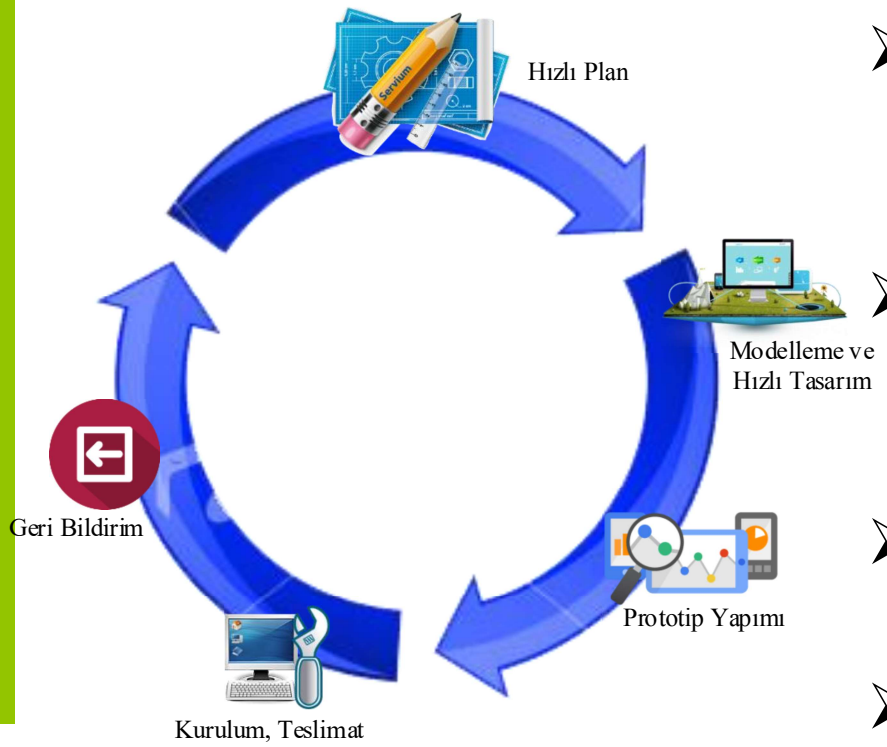
- Gereksinim tanımlama fazında hızlıca yapılan kısmi gerçekleştirme.
- Gereksinimler netleştikçe prototip kullanıcı ihtiyaçlarına göre düzeltilir
- Müşteri memnun olana kadar düzeltmelere devam et

Prototipleme-Avantajları



- Kullanıcı sistem gereksinimlerini görebilir
- Karmaşa ve yanlış anlaşılmaları engeller
- Yeni ve beklenmeyen gereksinimler netleştirilebilir
- Risk kontrolü sağlanır

Prototipleme-Dezavantajları

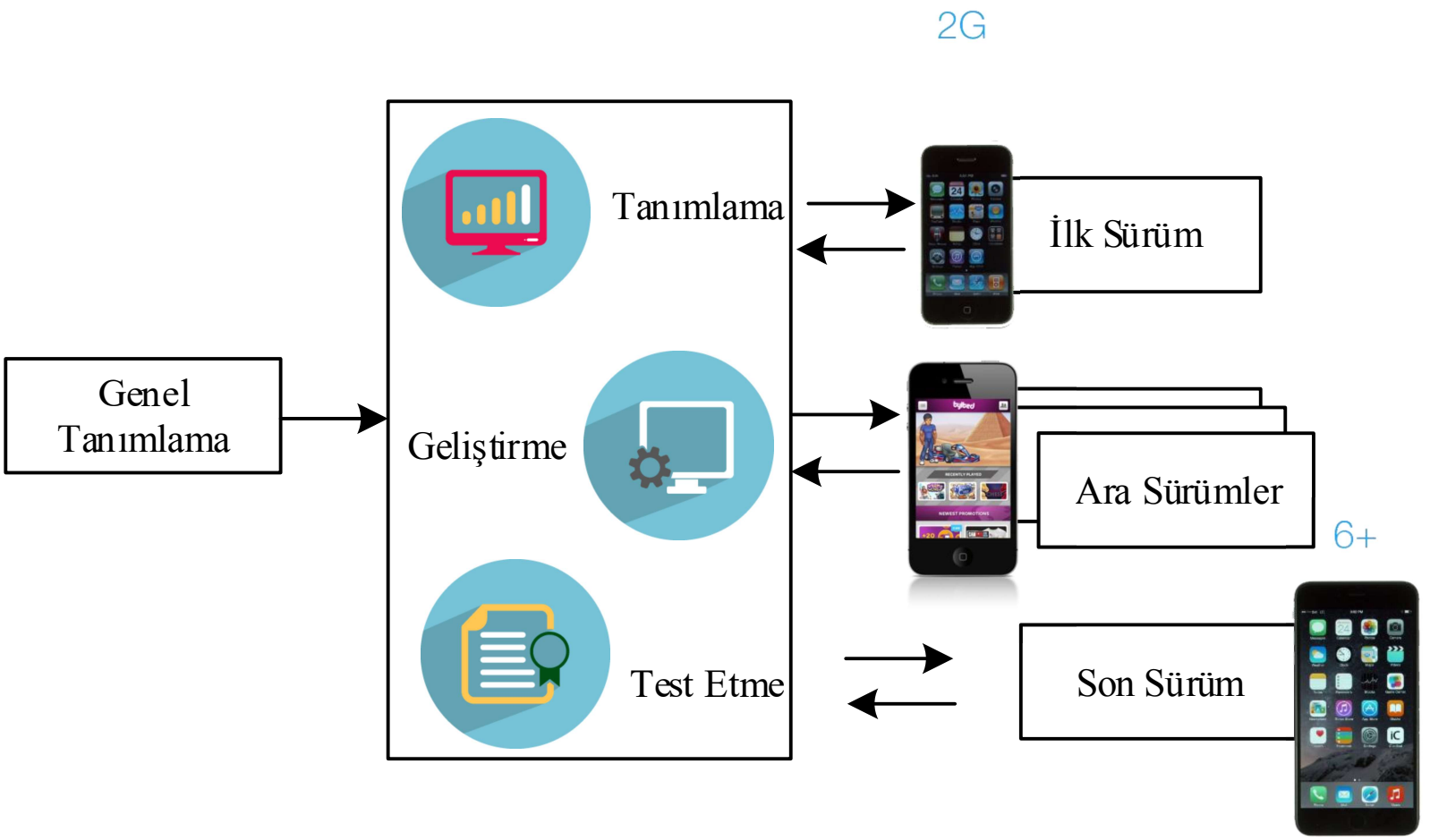


- Belgelendirmesi olmayan hızlı ve kirli (quick and dirty) prototipler
- Prototip hedefleri net değilse kod hackleme ya da jenga başlar
- Düzeltme aşaması atlanırsa, düşük performansa yol açar
- Müşteri prototipten de son ürün gibi görünüm ve etki bekler

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 içermektedirler.
- Pilot uygulama kullan, test et, güncelle diğer birimlere taşı.
- Modelin başarısı ilk evrimin başarısına bağımlıdır.

Evrimsel Geliştirme Süreç Modeli



Örnek

- Çok birimli banka uygulamaları.
- Önce sistem geliştirilir ve Şube-1'e yüklenir.
- Daha sonra aksaklıklar giderilerek geliştirilen sistem Şube-2'ye yüklenir.
- Daha sonra geliştirilen sistem Şube-3'e,…. yüklenir.
- Belirli aralıklarla eski şubelerdeki güncellemeler yapılır.

Evrimsel Geliştirme Süreç Modeli

İki çeşit evrimsel geliştirme vardır:

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”

Atılacak prototipleme (throw-away prototyping)

- Hedef: Sistem gereksinimlerini anlamak
- Tam anlaşılmamış gereksinimlerle başlar

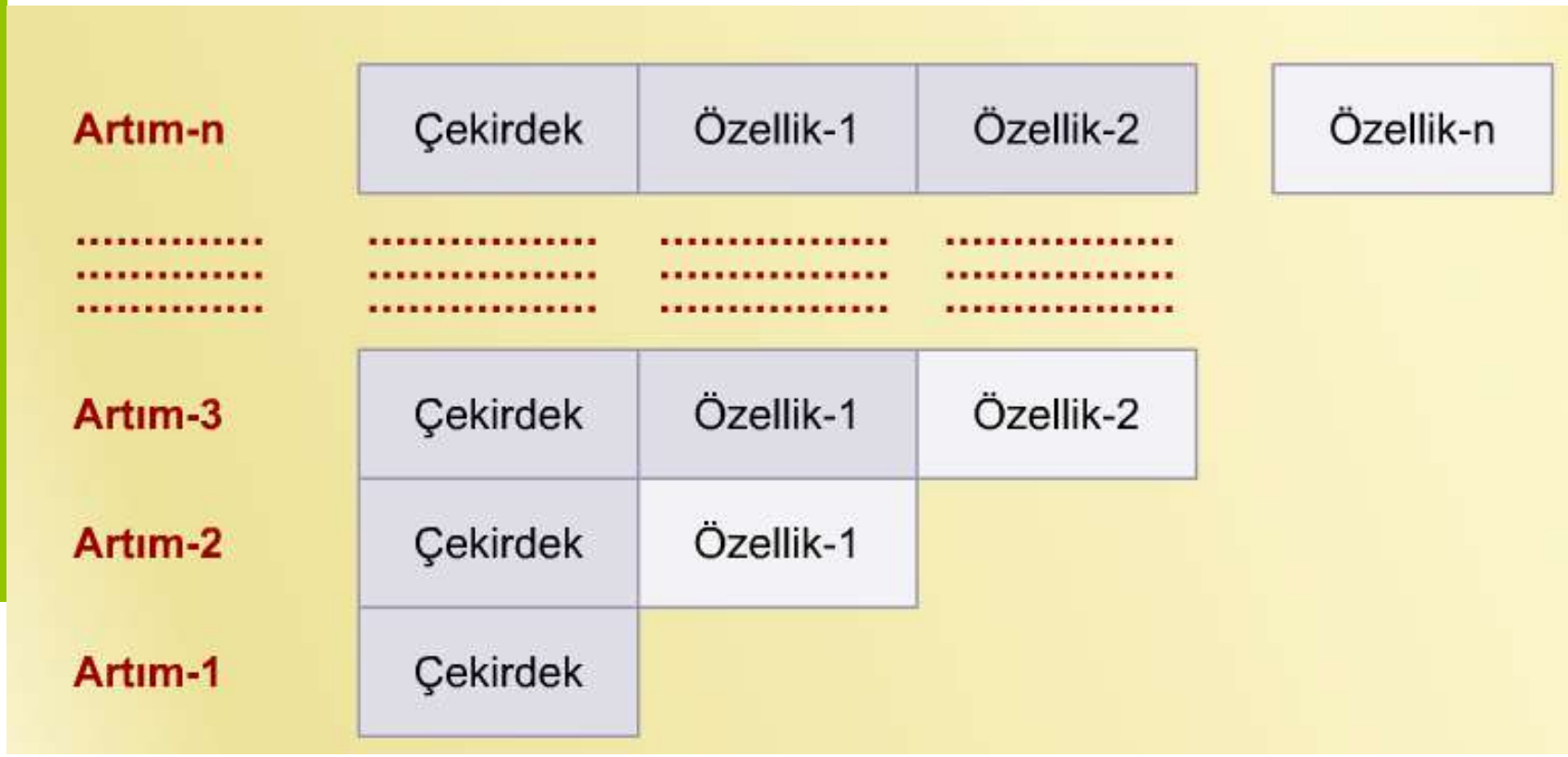
Aksaklığı

- Değişiklik denetimi
- Konfigürasyon Yönetimidir
  - Sürüm Yönetimi
  - Değişiklik Yönetimi
  - Kalite Yönetimi

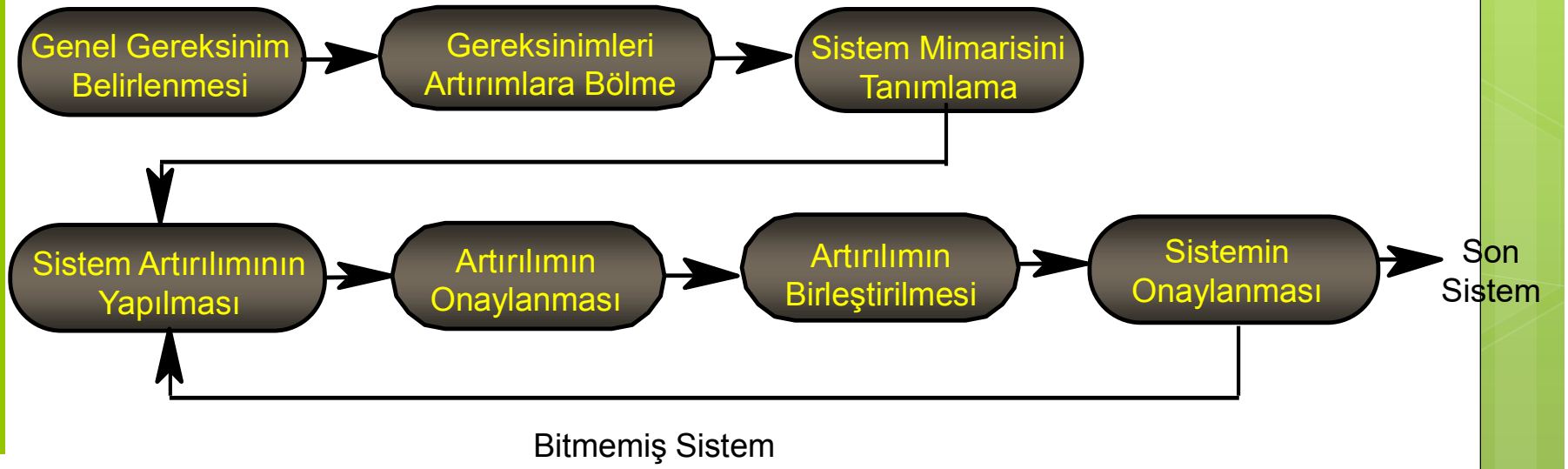
Artırımsal Geliştirme Süreç Modeli

- Üretilen her yazılım sürümü birbirini kapsayacak ve giderek artan sayıda işlev içerecek şekilde geliştirilir.
- Öğrencilerin bir dönem boyunca geliştirmeleri gereken bir programlama ödevinin 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.

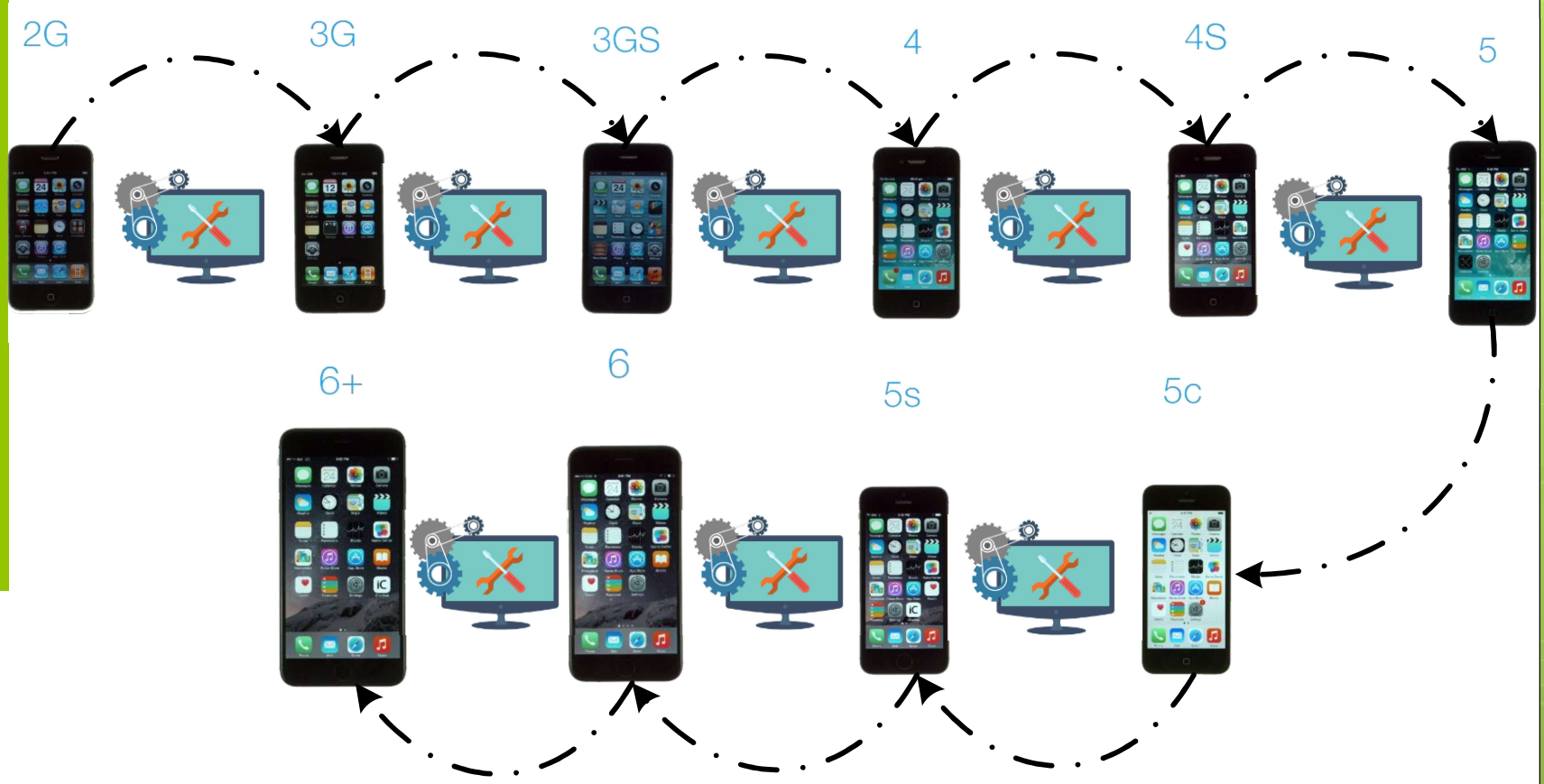
Artırımsal Geliştirme Süreç Modeli



Artırımsal Geliştirme Süreç Modeli



Artırımsal Geliştirme Süreç Modeli



Araştırma Tabanlı Süreç Modeli

- Yap-at prototipi olarak ta 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.

Araştırma Tabanlı Süreç Modeli



Örnek

- 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 bütünü.
- Bir metodoloji,
  - bir süreç modelini ve
  - belirli sayıda belirtim yöntemini içerir
- Günümüzdeki metodolojiler genelde Çağlayan ya da Helezonik modeli temel almaktadır

Bir Metodolojide Bulunması Gereken Temel Bileşenler (Özellikler)

- Ayrıntılandırılmış bir süreç modeli
- Ayrıntılı süreç tanımları
- İyi tanımlı üretim yöntemleri
- Süreçlerarası arayüz tanımları
- Ayrıntılı girdi tanımları
- Ayrıntılı çıktı tanımları
- Proje yönetim modeli
- Konfigürasyon yönetim modeli
- Maliyet yönetim modeli
- Kalite yönetim modeli
- Risk yönetim modeli
- Değişiklik yönetim modeli
- Kullanıcı arayüz ve ilişki modeli
- Standartlar

Bir Metodolojide Bulunması Gereken Temel Bileşenler

- Metodoloji bileşenleri ile ilgili olarak bağımsız kuruluş (IEEE, ISO, vs.) ve kişiler tarafından geliştirilmiş çeşitli standartlar ve rehberler mevcuttur.
- Kullanılan süreç modelleri ve belirtim yöntemleri zaman içinde değiştiği için standart ve rehberler de sürekli güncellenmektedir.
- Bir kuruluşun kendi metodolojisini geliştirmesi oldukça kapsamlı, zaman alıcı ve uzmanlık gerektiren bir faaliyet olup, istatistikler yaklaşık 50 kişi/ay'lık bir iş gücü gerektirdiğini göstermektedir.

Bir Metodoloji Örneği

- Yourdan Yapısal Sistem Tasarımı Metodolojisi.
- Kolay uygulanabilir bir model olup, günümüzde oldukça yaygın olarak kullanılmaktadır.
- Çağlayan modelini temel almaktadır.
- Bir çok CASE aracı tarafından doğrudan desteklenmektedir.

Yourdon Yapısal Sistem Tasarım Metodolojisi

Aşama	Kullanılan Yöntem ve Araçlar	Ne için Kullanıldığı	Çıktı
Planlama	Veri Akış Şemaları, Süreç Belirtimleri, Görüşme, Maliyet Kestirim Yöntemi, Proje Yönetim Araçları	Süreç İnceleme Kaynak Kestirimi Proje Yönetimi	Proje Planı
Analiz	Veri Akış Şemaları,Süreç Belirtimleri,Görüşme,Nesne ilişki şemaları ,Veri	Süreç Analizi Veri Analizi	Sistem Analiz Raporu
Analizden Tasarıma Geçiş	Akışa Dayalı Analiz, Süreç belirtimlerinin program tasarım diline dönüştürülmesi, Nesne ilişki şemalarının veri tablosuna dönüştürülmesi	Başlangıç Tasarımı Ayrıntılı Tasarım Başlangıç Veri tasarımı	Başlangıç Tasarım Raporu
Tasarım	Yapısal Şemalar, Program Tasarım Dili, Veri Tabanı Tabloları	Genel Tasarım Ayrıntılı Tasarım Veri Tasarımı	Sistem Tasarım Raporu

Süreç Modeli	Örnek Proje
1. Gelişigüzel Model	Öğrenci Ders Projeleri
2. Barok Model	Kullanım Kılavuzu Gerektiren Basit Projeler
3. Çağlayan Modeli	Hastane Sistemleri, Kontrol Sistemleri
4. V Modeli	CRM (Müşteri İlişkileri Yönetimi) Projeleri
5. Helezonik Model	Askeri Savunma Yazılım Sistemleri
6. Arttırımsal Geliştirme	Öğrenci Bitirme Projeleri
7. Prototipleme	Yüksek Maliyetli Projelerin İlk Aşamasının Bilgisayar Destekli Gerçekleştirilmesi
8. Evrimsel Geliştirme	Çok Birimli Organizasyonlar İçin (Bankacılık Uygulamaları)
9. Araştırma Tabanlı Model	En Hızlı Çalışan Asal Sayı Test Programı, En Büyük Asal Sayıyı Bulma Programı, Satranç Programı

İçindekiler

1 Günümüzde Yazılım Projelerinin Durumu.....3

2 Çevik Yazılım Geliştirme Yöntemi.....9

3 Geleneksel Model vs. Agile..... 17

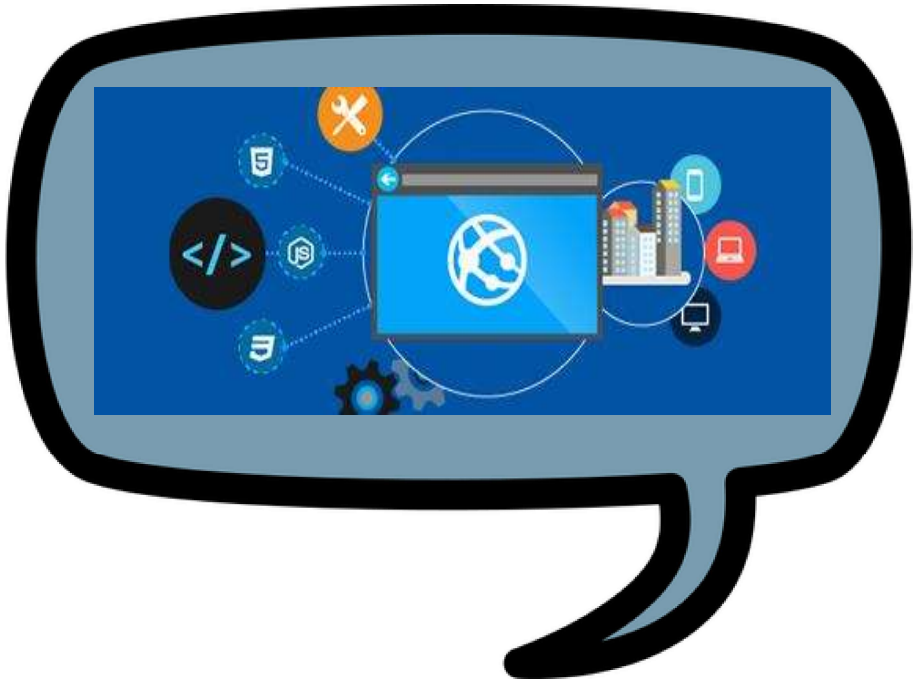
4 Değerlendirme.....26

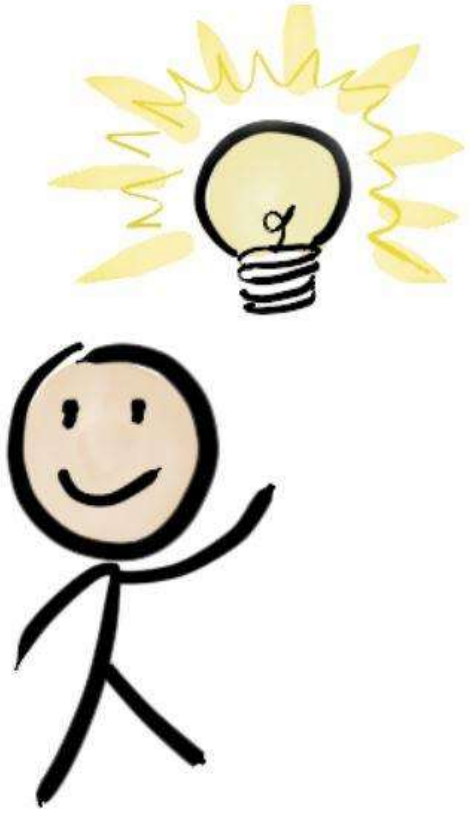
5 Çevik Yazılım Şemsiyesi.....28

6 Scrum Modeli.....29

1.Günümüzde Yazılım Projelerinin Durumu

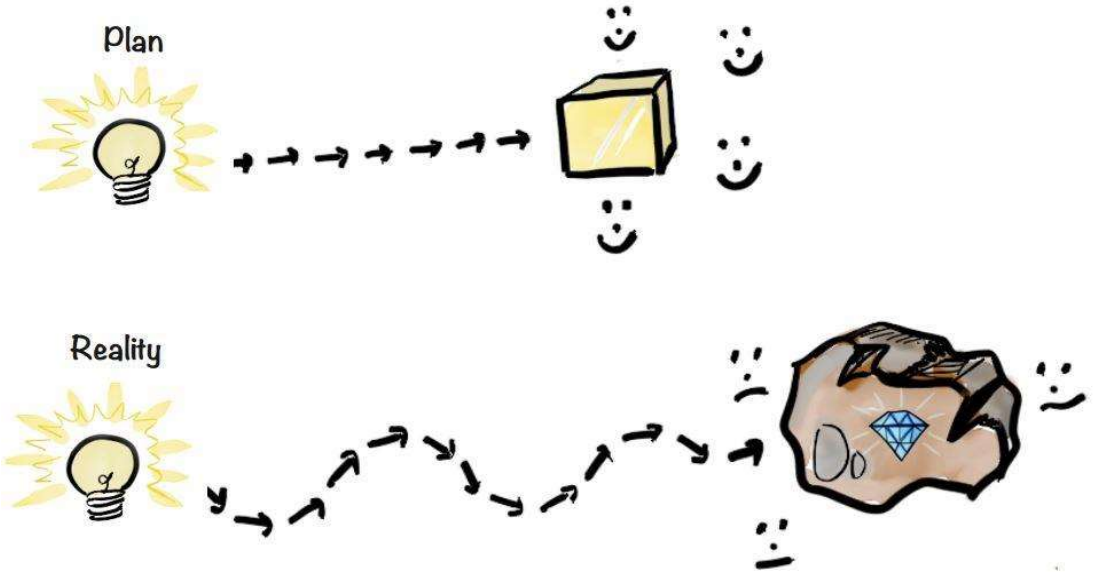
- Birçok proje harika bir fikir ile başlar!





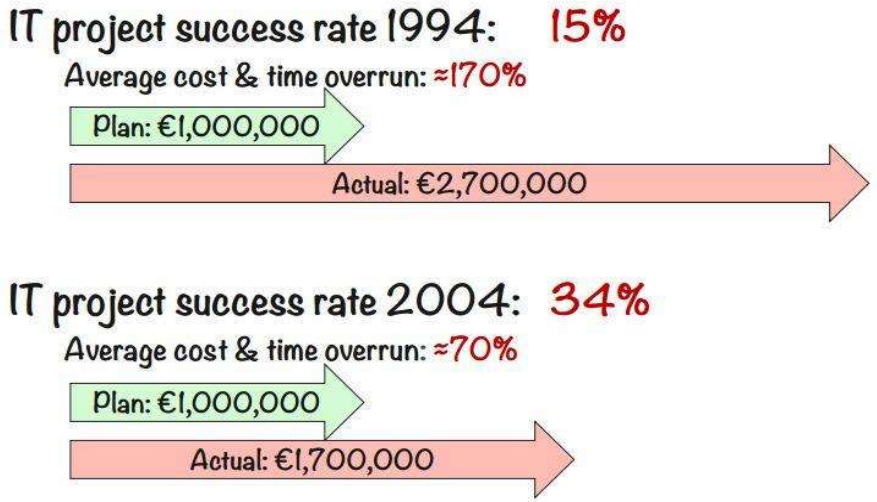
1.Günümüzde Yazılım Projelerinin Durumu

- Bu projelerin büyük bir kısmının başarısız olması muhtemeldir!



1.Günümüzde Yazılım Projelerinin Durumu

- Birçok Bilgi Teknolojisi projesi başarısız olmuş veya gecikmiştir. The Standish Group, 10 yıl içerisinde 40.000'den fazla proje üzerinde çalışmıştır.



1.Günümüzde Yazılım Projelerinin Durumu

Ülkemizde durum nasıl?

Durum	Oran
Tam başarılı	%4-5
Kısmen başarılı	%45-50
Çöpe gidenler	%50

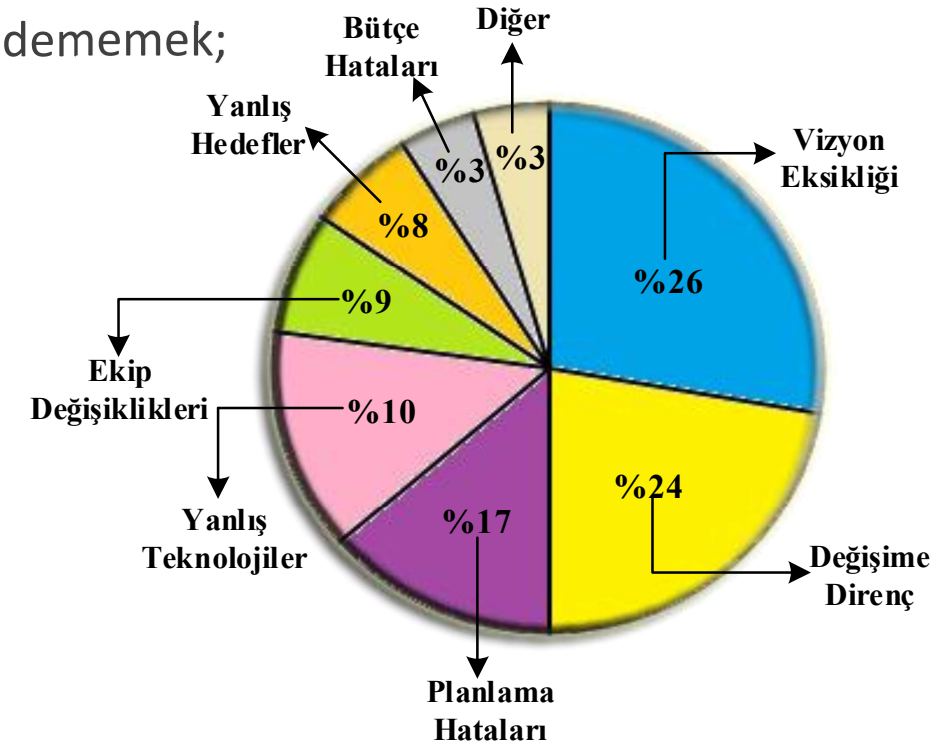
Agile Turkey

Örneğin, ülkemizde geliştirilen projelerin başarıya ulaşmasına katkı sağlamak amacıyla, Araştırma Destek Programları Başkanlığı (ARDEB) tarafından desteklenen projelerin çıktı, sonuç ve etkilerini nicelik ve nitelik olarak artırmak amacıyla yüksek başarı ile sonuçlanan projelerin yürütücü ve araştırmacılarını ödüllendirmek için TÜBİTAK tarafından belirlenen ölçütler ve değerlendirme yöntemine göre hesaplanarak, proje ekibine (yürütücü ve araştırmacılara) TÜBİTAK Proje Performans Ödülü (PPÖ), denilen bir teşvik ödülü verilmektedir.

1.Günümüzde Yazılım Projelerinin Durumu



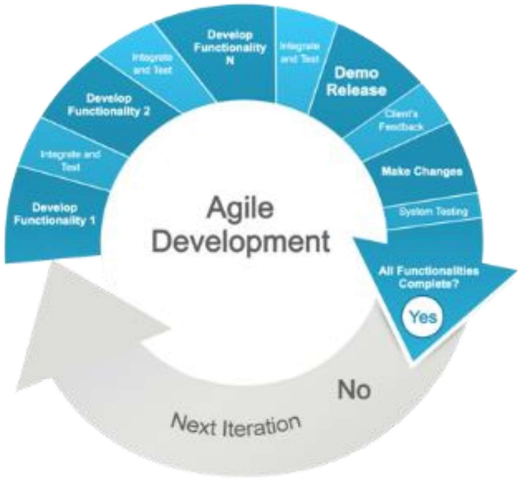
- Başarısızlığın ana sebepleri:
  - Müşterinin isteklerini doğru analiz edememek;
  - Proje için uygun ekibi kuramamak;
  - Yanlış teknoloji ve mimari seçimleri;
  - Geleneksel yöntemlerin eksiklikleri;
  - Müşteriyle iletişimden kaçınmak vs.



1.Günümüzde Yazılım Projelerinin Durumu

Peki ne yapmalıyız?





2.Çevik Yazılım Yöntemi



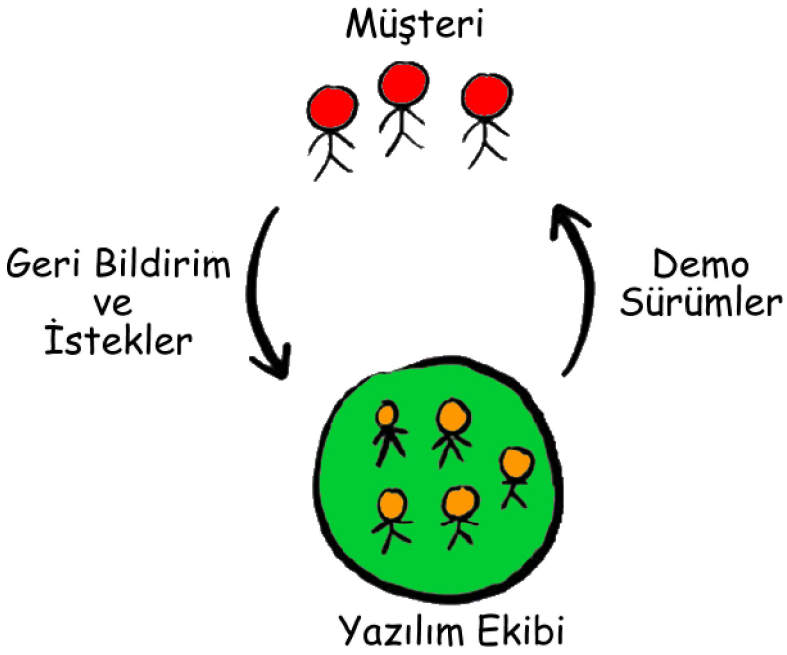
- ***Tekrarlamalı*** ve ***artımsal*** bir ürün geliştirme yöntemidir.
  - **Bireyler ve etkileşimi**, süreç ve araca tercih eder.
  - **Çalışan bir yazılımı**, detaylı ürün belgelendirmeye tercih eder.
  - **Müşteri ile işbirliğini**, sözleşmedeki kesin kurallara tercih eder.
  - **Değişikliklere uyum sağlayabilmeyi**, belirli bir plana tercih eder.

2.Çevik Yazılım Yöntemi



- 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.

2.Çevik Yazılım Yöntemi

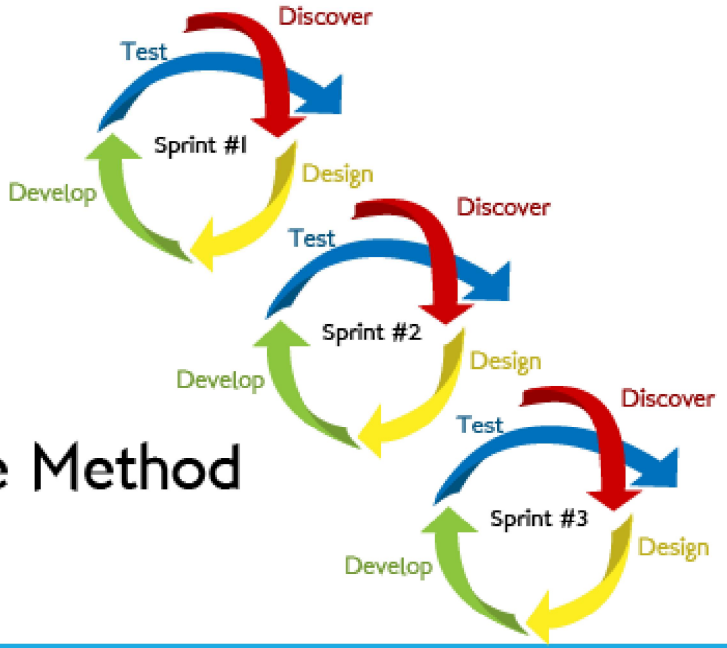


- Çevik yazılım metodu, kısa vadeli planlar ve küçük parçalar halinde yazılımın geliştirilmesini ön görür. Yazılımın geliştirilmesindeki geri dönüş (feedback) ve değişikliklere uyum sağlamak son derece önemlidir. Her yapılan yineleme yazılımı hedeflenen adıma bir adım daha yakınlaştırır. İstenilen sonuca ulaşmak adına birden çok yineleme gereklidir.

2.Çevik Yazılım Yöntemi

- Örnek:



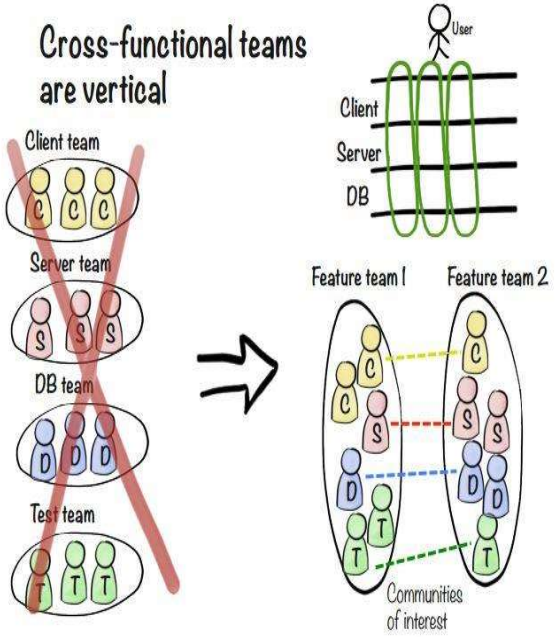


2.Çevik Yazılım Yöntemi

- 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 ?**

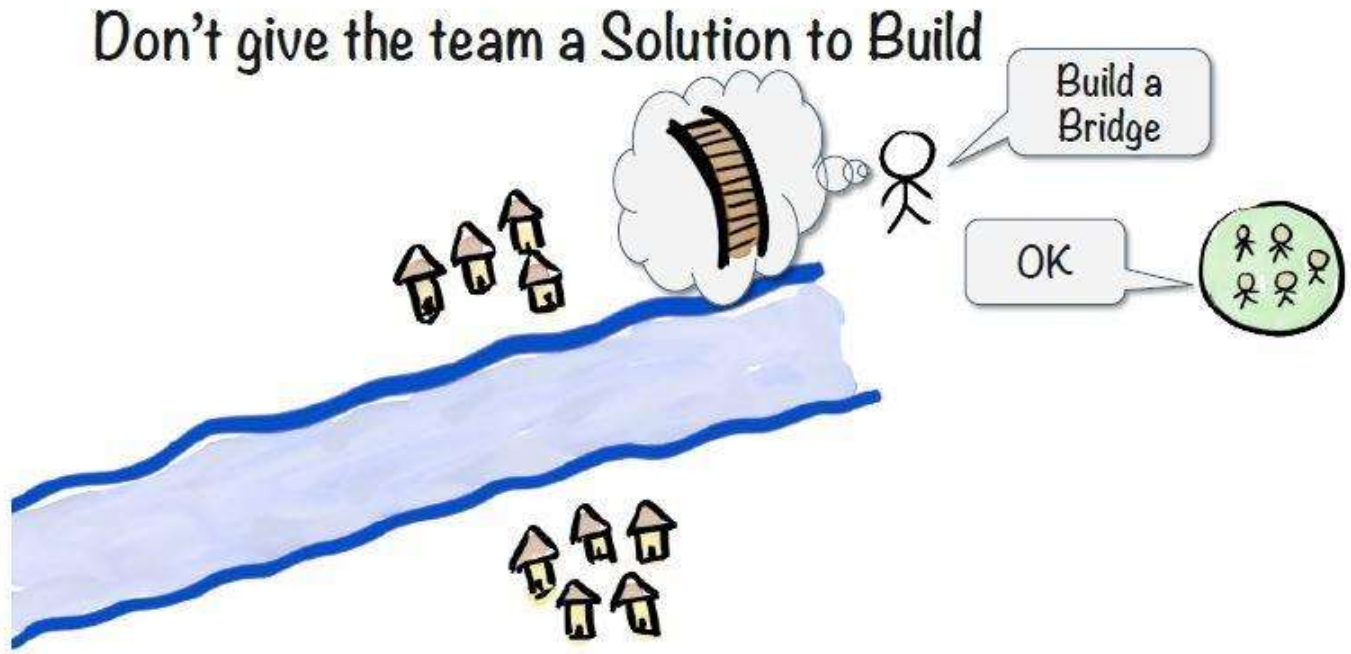


2.1.Çevik Model Takımları



- Biraraya gelmiş,
- Kendi kendilerine organize olan,
- Çapraz fonksiyonlu,
- İşine odaklanmış,
- Hedefleri net olan,
- Teslim edilebilecek düzeyde ürün ortaya koyabilen
- Küçük(3-7 kişilik) gruplar.

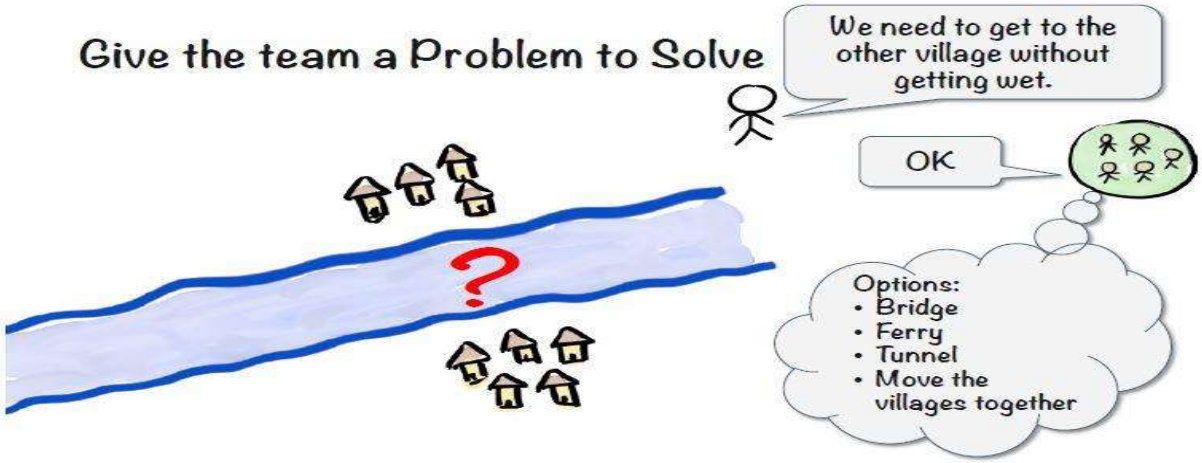
2.1.Çevik Model Takımları



Takımlara çözümü söylemeyin!

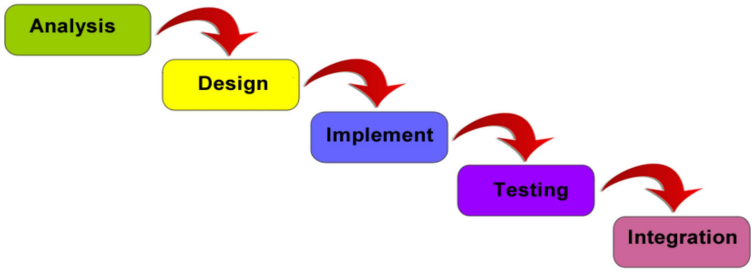
2.1.Çevik Model Takımları

Sorunu söyleyin, onlar çözümü üretsin!



3. Geleneksel Model vs. Agile

- Çağlayan modeli 2008 yılında dahi geçerliliğini koruyan bir modeldir ve çevik modellemeden farklılık gösterir. Bu model yazılım projesini baştan sona planlar. Gelişim, sunulabilir işler açısından ölçülür: talep açıklamaları, tasarım dokümanları, test planları, kod incelemeleri vb. Bu durum belli aralıklara bölünmeye uygun değildir ve ilerideki değişikliklere uyum gösterilemez.



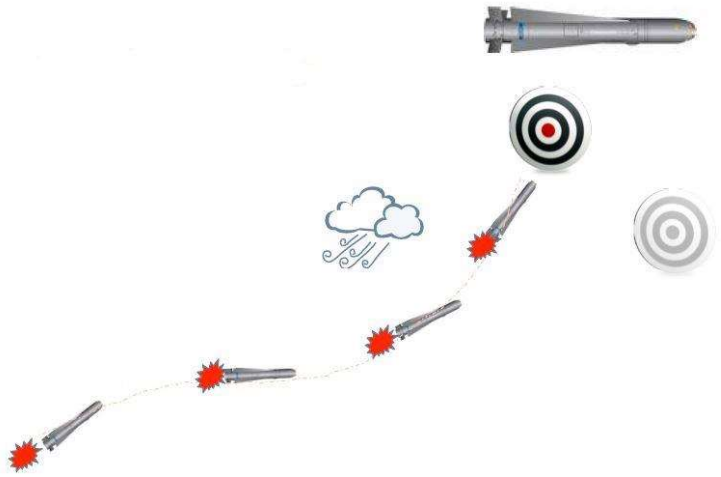
Waterfall Model

3. Geleneksel Model vs. Agile



- Geleneksel Yöntemler
- Müşteriler ne istediğini iyi bilir.
- Geliştiriciler neyi, ne şekilde üreteceklerini iyi bilir.
- Bu yol boyunca hiç birşey değişmeyecektir.

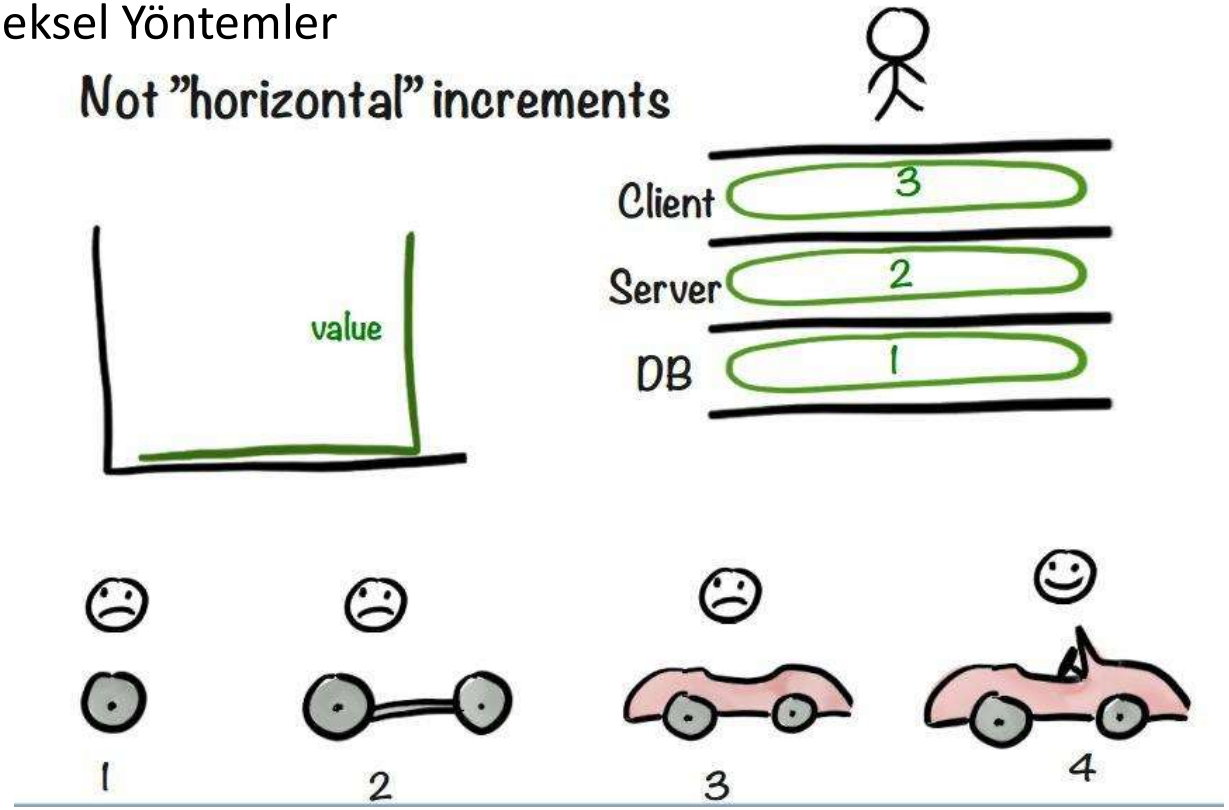
3. Geleneksel Model vs. Agile



- Çevik Yöntemler
- Müşteriler ne istediğini keşfeder.
- Geliştiriciler neyi nasıl üreteceğini keşfeder.
- Bu yol boyunca bir çok değişiklik yapılabilir.

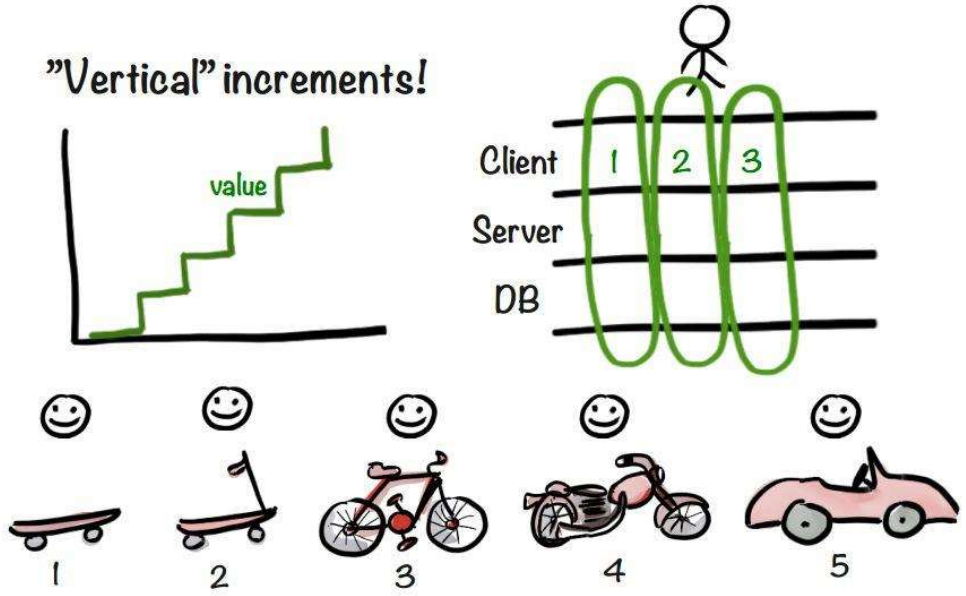
3. Geleneksel Model vs. Agile

Geleneksel Yöntemler

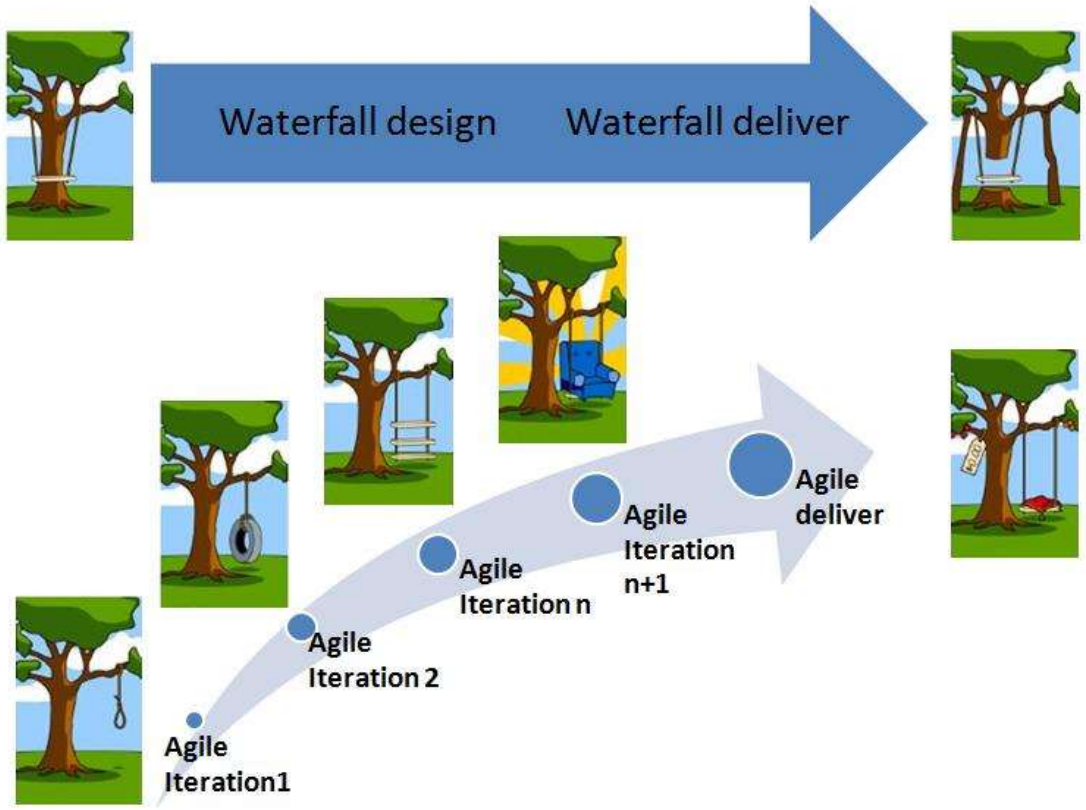


3. Geleneksel Model vs. Agile

Çevik Yöntemler

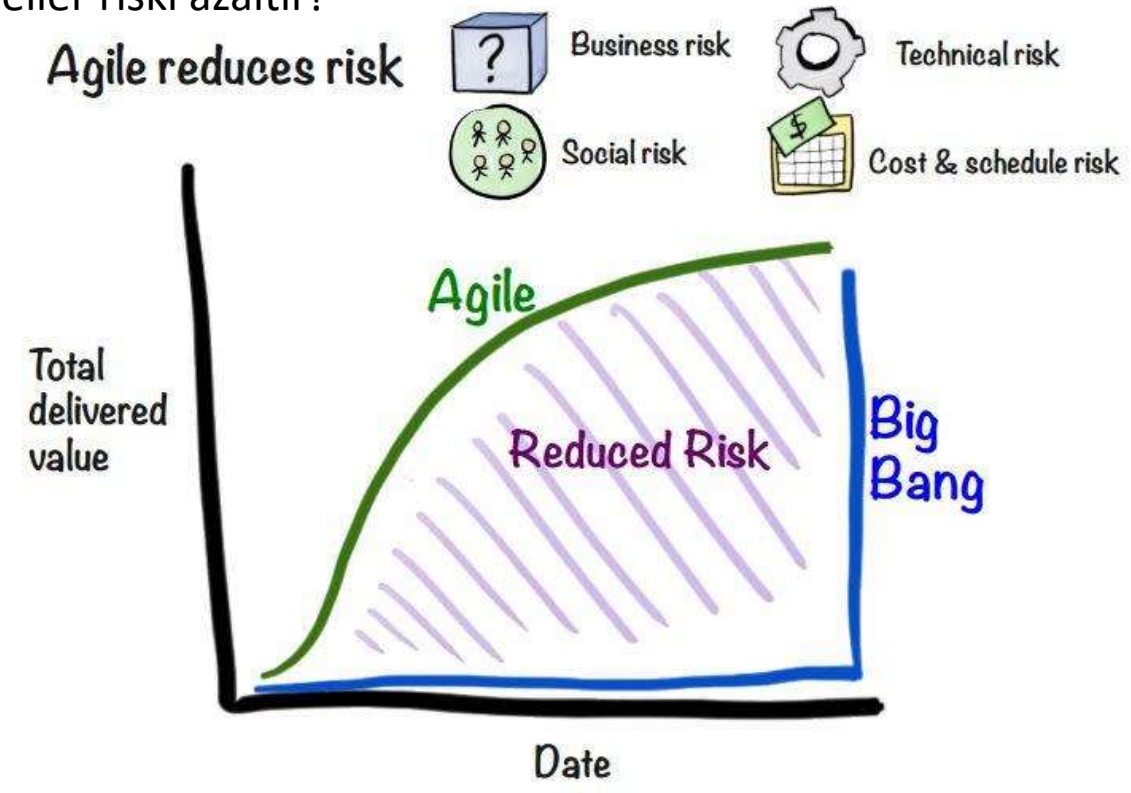


3. Geleneksel Model vs. Agile

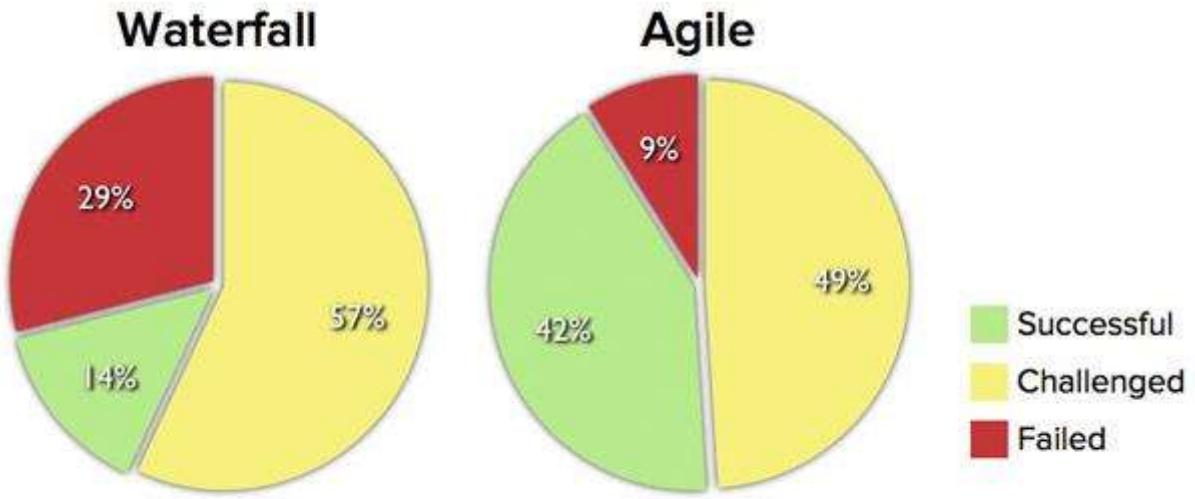


3. Geleneksel Model vs. Agile

Çevik modeller riski azaltır!



3. Geleneksel Model vs. Agile

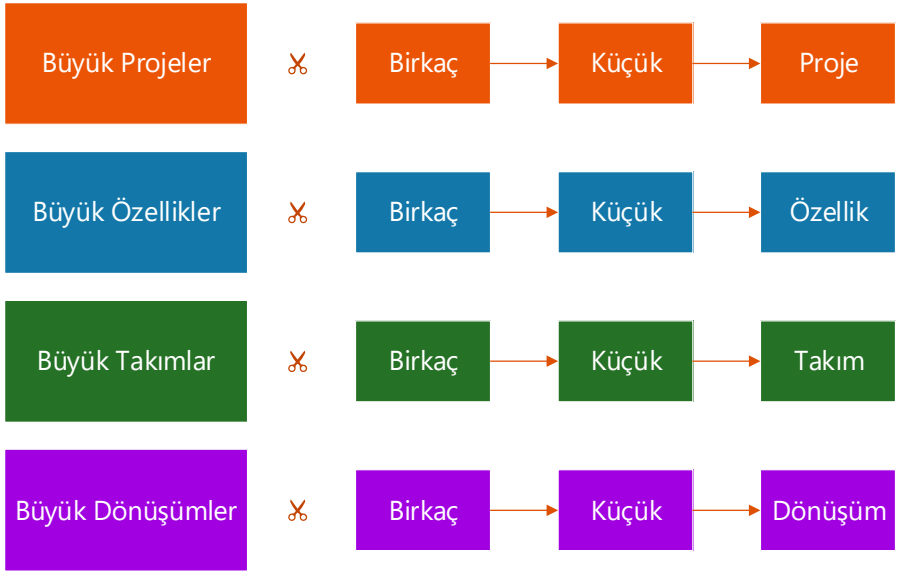


Source: The CHAOS Manifesto, The Standish Group, 2012.

3. Geleneksel Model vs. Agile

Ölçüm	Çevik Modelleme	Çağlayan Modeli
Planlama ölçeği	Kısa dönemlik	Uzun dönemlik
Müşteri ile geliştirici arasındaki mesafe	Kısa	Uzun
Özelleştirme ve uygulama arasındaki zaman	Kısa	Uzun
Sorunları keşfetmek için zaman	Kısa	Uzun
Proje tamamlanma riski	Düşük	Yüksek
Değişikliklere uyum yeteneği	Yüksek	Düşük

4. Değerlendirme



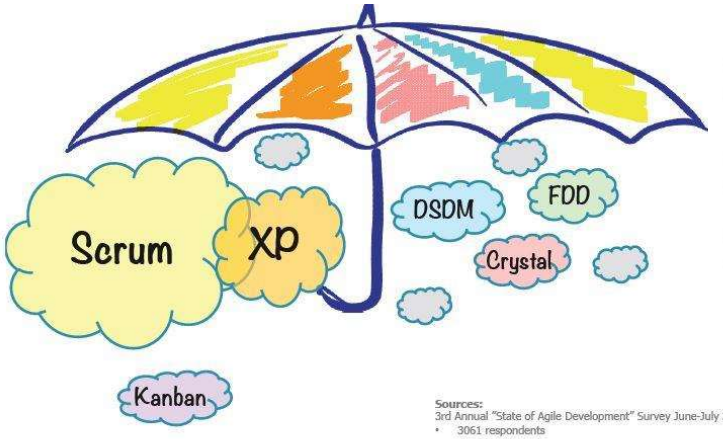
Sonuç

- Büyüklük kötüdür, onu parçalara ayırın. Böylece daha başarılı projeler geliştirebilirsiniz.

4. Değerlendirme

- 3 somut değişiklik yapın!
- 1. Gerçek takımlar oluşturun
  - Küçük, çapraz fonksiyonlu, kendi kendine organize olabilen
- 2. Sık sık teslimat yapın
  - Normal olarak ortalama her 3 haftanın sonunda
  - Ek olarak projenin tüm çeyreklerinin sonunda
- 3. Gerçek kullanıcıları dahil edin.
  - Takım ve kullanıcılar arasında doğrudan ve hızlı geri dönüşler

5. Çevik Yazılım Şemsiyesi



Sources:
3rd Annual "State of Agile Development" Survey June-July 2008
- 3061 respondents
- 80 countries

FDD: Feature-Driven Development

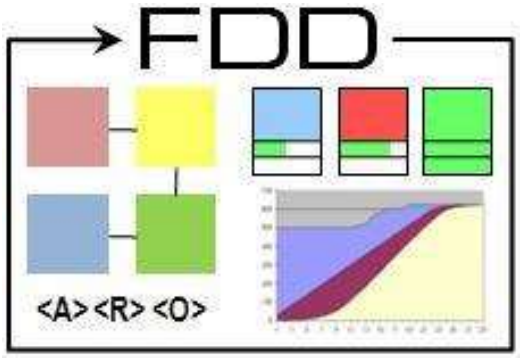
RUP : Rational Unified Process

DSDM : Dynamic System Development Method

FDD TANIMI

- **FDD, Avustralyalı Jeff De Luca tarafından geliştirilmiş ve Singapur projesinde beraber çalıştıkları Peter Coad tarafından modifiye edilmiştir. Singapur projesi 50 kişi ile 15 ayda tamamlanan bir proje olmuştur. Daha sonra FDD'nin uygulandığı başka bir proje ise 250 kişi ile 18 ayda tamamlanmıştır. FDD değişik boyutlara büyüyebilen, tekrarlanabilir bir süreçtir. Aşağıdaki noktalara odaklanır.**

1. **Sistemi hazırlamak için gereken sistem büyüyebilir olmalıdır. Büyük projeler için de kullanılabilir olmalıdır.**
2. **Basit iyi tanımlanmış sistem iyi çalışır.**
3. **Süreç adımları basit olmalıdır.**
4. **İyi süreç arka plana saklanır ve insanlar sonuçlara odaklanabilir.**
5. **Kısa, iteratif, özellik yaklaşımlı yaşam döngüleri en iyi sonucu verir**



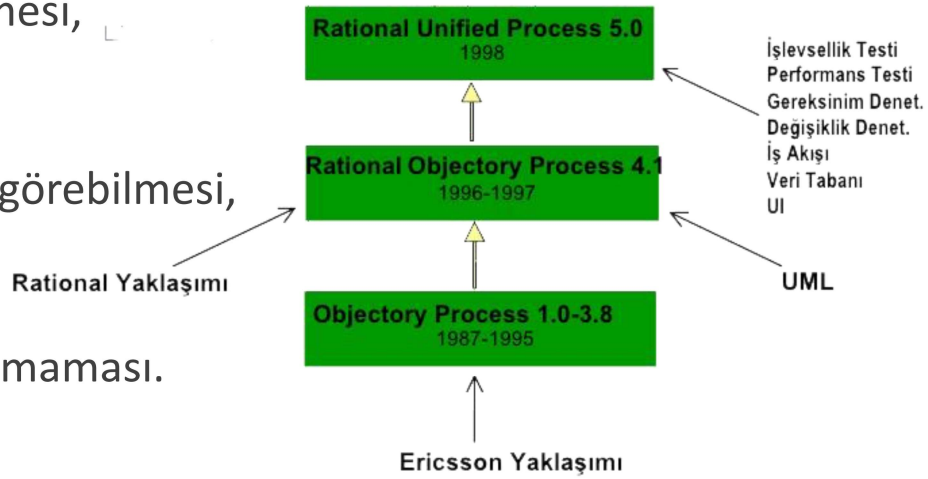
RUP ("Rational Unified Process")

- 2003 yılından beri IBM'in bir bölümü tarafından oluşturulan bir iteratif yazılım geliştirme süreci çerçevesidir. Başarısız bir yazılımdaki sorunların aşılıp başarılı yazılım oluşturmak için gerekli adımları saptayarak oluşturulmuş bir süreçtir.

- Başarısız bir yazılımdaki özelliklerini yazımızın devamında okuyabilirsiniz. RUP şirketlere yazılım geliştirme aşamasında bir yön sağlayar. RUP use-case ve nesne teknolojileri tabanlı; tekrarlanan (iterative) yazılım geliştirme ve iş modelleme yöntemidir. RUP'un verebileceği özellikler şunlardır;

1. Müşteriyi ve yazılımcıyı organize edebilmesi,
2. Standart tanımlı adımları olması,
3. Oluşacak yazılımdaki sık değişiklikleri öngörebilmesi,
4. Basit olması,
5. Proje yönetim aktivitelerinin çok fazla olmaması.

RUP'un Gelişimi



Uç Programlama (Extreme Programming XP) Nedir?

➢ **Uç Programlama (XP)**, yazılım geliştirme süreci boyunca son derece *kaliteli olmak koşuluyla* **çalıştırılabilir kod** üretmeye odaklanmış bir yazılım geliştirme metodolojisidir. Yazılım geliştirme sürecinin en temel, en önemli ve final çıktısı ya da ürünü çalıştırılabilir kod olduğundan, XP metodolojisi sürecin en başından itibaren çalıştırılabilir kodu sürecin merkezinde tutmaktadır. İşte bu yüzden bu metodolojinin adı XP'dir.

Uç Programlama'nın özünde aşağıdaki uygulamalar yer alır:

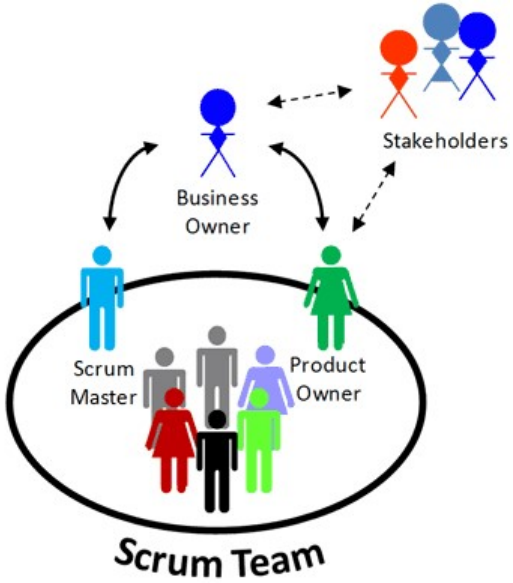
- Planlama
- Sık ve küçük sürümler
- Basit tasarım:
- Önce test
- Refactor etme
- Haftada 40 saat çalışma
- Müşteriyle yakın iletişim
- Kodlama standartları

6. Scrum Modeli



Scrum Süreci

6. Scrum Modeli



- Scrum Takımı: Ürün Sahibi, Geliştirme Ekibi ve Scrum Master'dan oluşur. Takım kendi kendini örgütler. Böylece kendi içerisinde uyum içinde olan takımlar daha başarılı sonuçlar alırlar. Scrum takım modeli esneklik, yaratıcılık ve verimliliği optimize etmek için tasarlanmıştır.

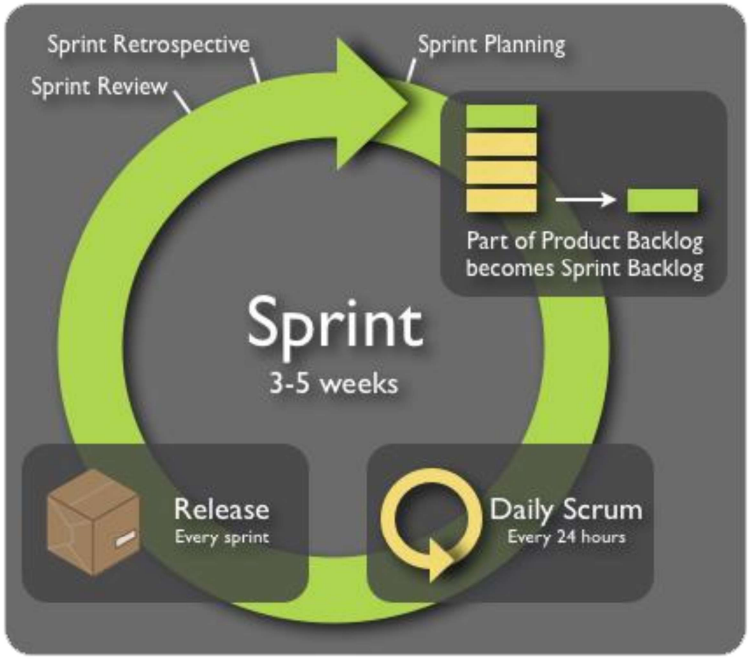
6. Scrum Modeli



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.

6. Scrum Modeli



- **Sprint**
- Belirli bir süreye sahiptir.
- Sonunda ortada değeri olan bir çıktı olmalıdır.
- Toplantılarla içerik belirlenir.
- Sprint süresi boyunca her gün toplantılar yapılır.

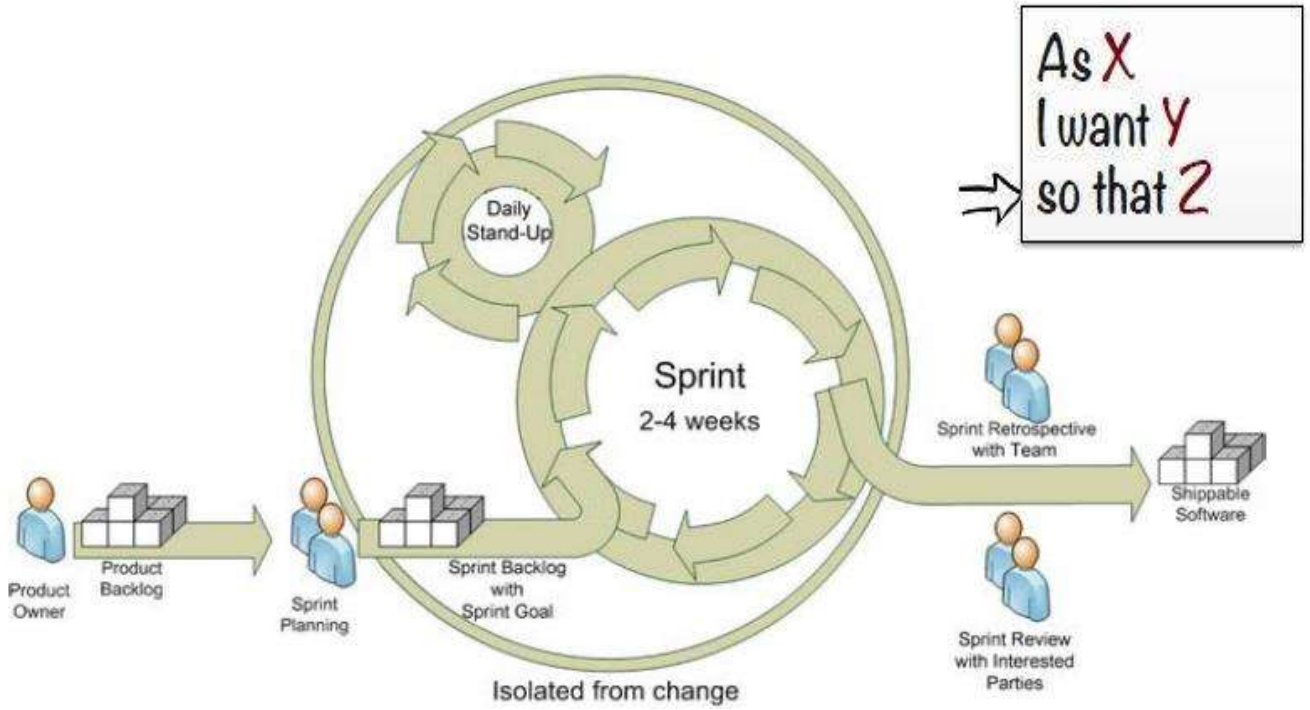
6. Scrum Modeli



Sprint Gösterimi

6. Scrum Modeli

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

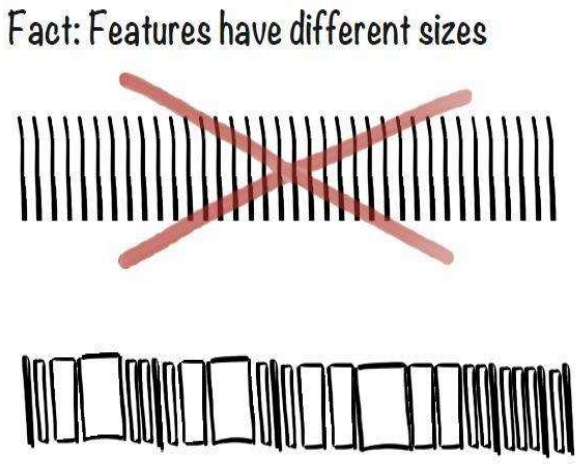


6. Scrum Modeli

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



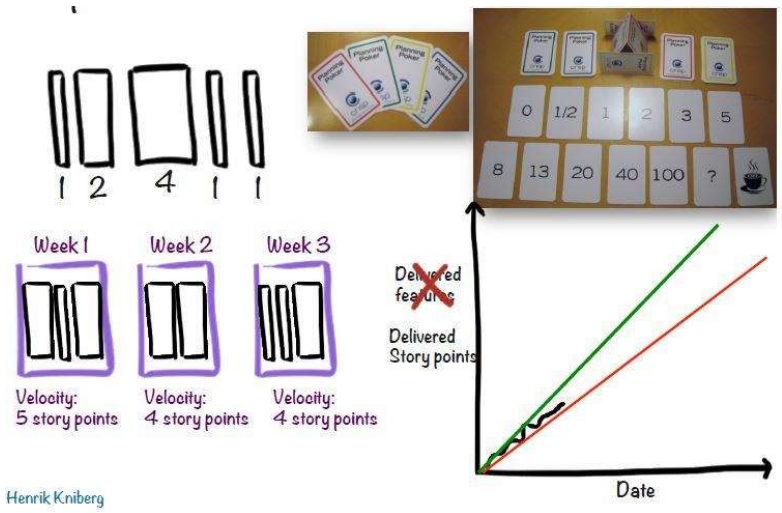
6. Scrum Modeli



- Her bir user story farklı bir boyuttadır. Somut olarak bakarsak, bir projedeki her bir gereksinim için gereken iş gücü ve zaman aynı değildir. Bu sebeple ürün backlogları sprintlere bölünürken, user storylerin boyut ve öncelikleri göz önünde bulundurulur. Örneğin bir sprint 3 user story içerirken diğeri daha küçük boyutlarda 5 user story içerebilir. Peki boyutları nasıl belirleyeceğiz?

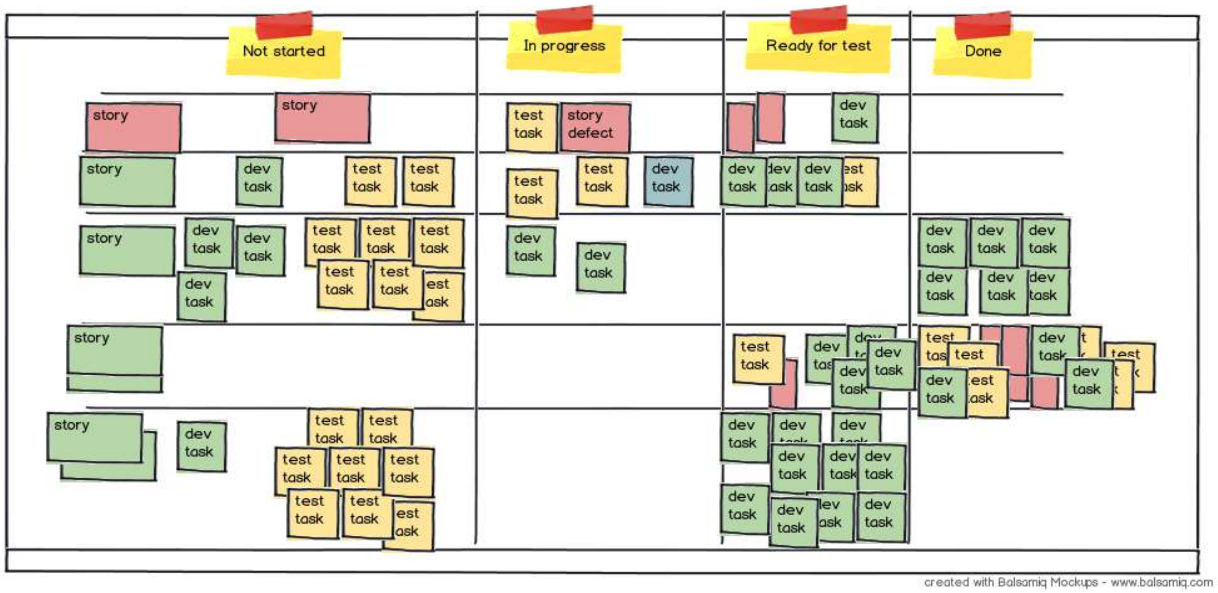
İş göründüğü gibi değil!

6. Scrum Modeli



- **Poker Kartları:** Scrum takım üyeleri bir araya gelir. Scrum master bir user story okur. Takımdaki her bir üye user story için uygun gördüğü poker kartlarından birini seçer. Herkes kartları seçtikten sonra tüm kartlar açılır ve değerlendirilir. Böylece herkesin ortak görüşü sonunda user story'lerin büyüklüğü belirlenir.

6. Scrum Modeli



Scrum Board

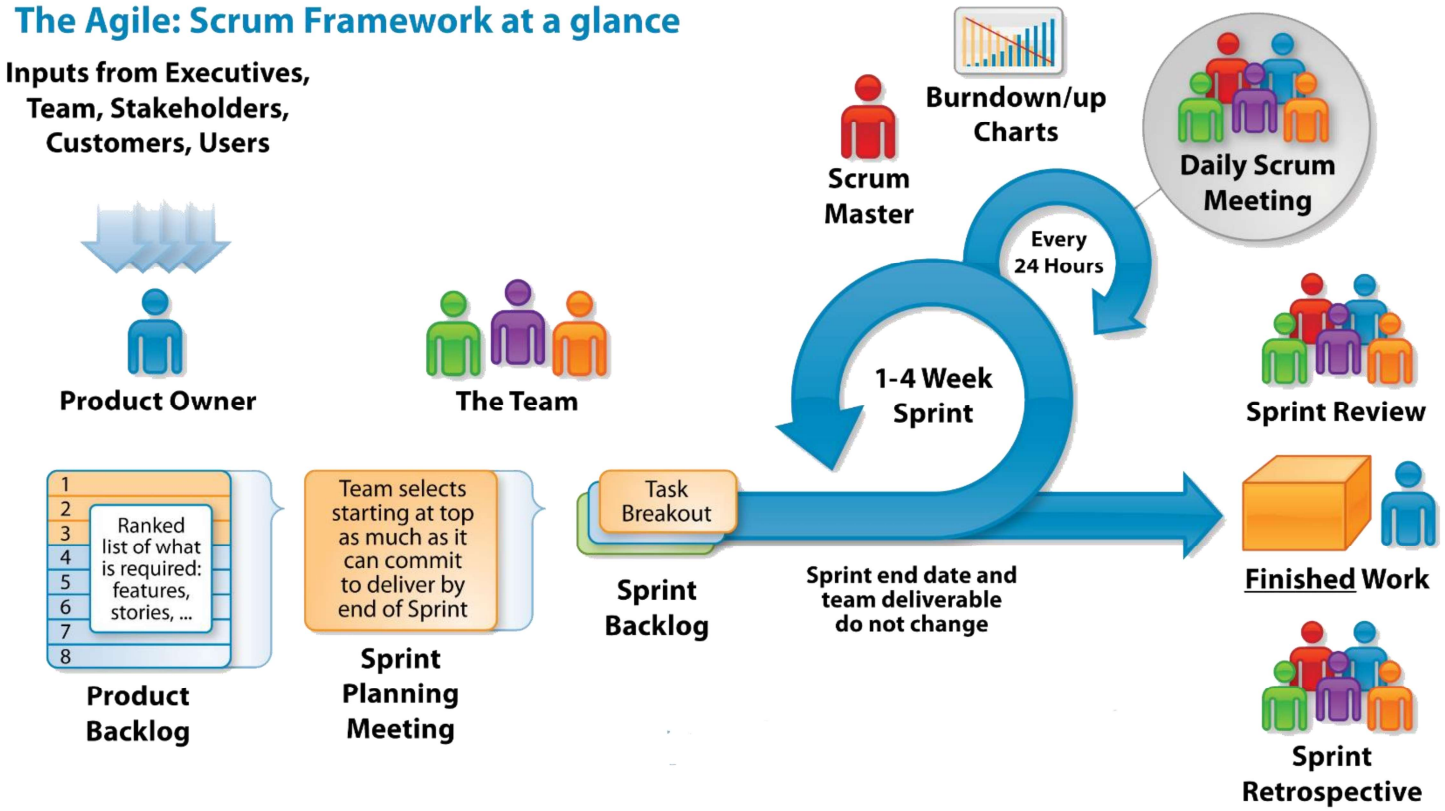
6. Scrum Modeli

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

Agile is...

Early delivery of business value

Less bureaucracy



Çalışma Soruları

1. Agile modelinin geliştirilmesine neden ihtiyaç duyulmuştur? Açıklayınız.

2. Agile takımları kimlerden oluşur ve iş paylaşımları nasıldır?

3. Agile modelinin diğer modellerden farkları nelerdir? Üzerinde durduğu temel noktalar nelerdir?

4. User story, backlog ve sprint kavramlarını açıklayınız.

5. Agile ile geliştirilmiş büyük yazılım projelerine örnekler veriniz.

6. Scrum haricindeki diğer çevik yazılım geliştirme yöntemlerini açıklayınız.

Kaynaklar

[1] Martin, Micah, and Robert C. Martin. *Agile principles, patterns, and practices in C#*. Pearson Education, 2006.

[2] Kniberg, Henrik. "Scrum and XP from the Trenches." *Lulu. com* (2007).

[3] Kniberg, Henrik. " What is agile" , 2013

[4] http://en.wikipedia.org/wiki/Agile_software_development

[5] http://www.mshowto.org/microsoft-visual-studio-team-foundation-server-nedir.html

[6] www.kurumsaljava.com/download/10/

[7]https://www.tubitak.gov.tr/tr/destekler/akademik/uygulamalar-ve-yonergeler/icerik-proje-performans-odulu-ppo-uygulamasi

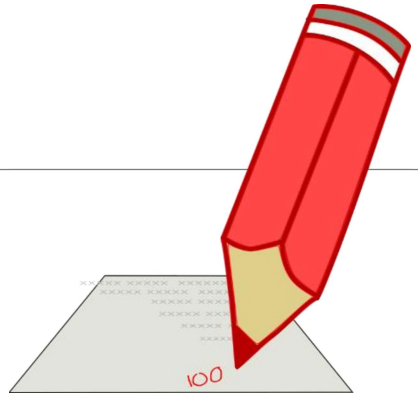
[8] http://images.slideplayer.biz.tr/8/2395426/slides/slide_27.jpg

[9] http://www.bayramucuncu.com/wp-content/uploads/2013/04/Ads%C4%B1z.png

[10] http://antasya.com/Images/is-alanlarimiz/yazilim-sistemleri/agile-method.png

[11] http://volkansel.com/wp-content/uploads/2014/07/agile-scrum.jpg

Ödev



Sorularınız



Agile Nedir?

Infografik Görsellerle



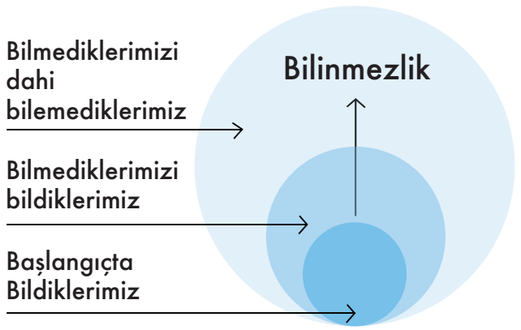
Bizler daha iyi yazılım geliştirme yollarını uygulayarak ve başkalarının da uygulamasına yardım ederek ortaya çıkartıyoruz.

Bu çalışmaların sonucunda:

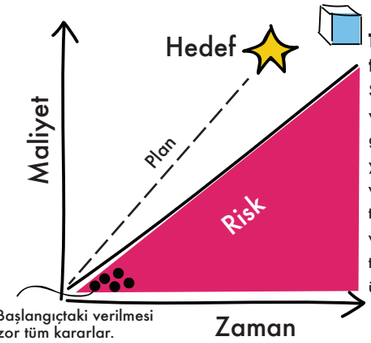
Bireyler ve etkileşimler süreçler ve araçlardan
Çalışan yazılım kapsamlı dokümantasyondan
Müşteri ile işbirliği müşteri ile pazarlıktan
Değişikliklere yanıt vermek bir planı takip etmekten
önce gelir.
2001 - Agile Manifesto

Version 1.0

Neden Agile

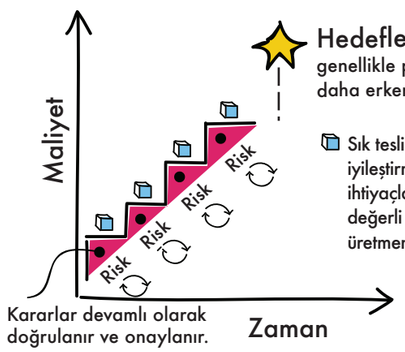


Waterfall Klasik Yöntemler



Tek seferde yapılan toplu teslimatlar
Sık test edilmeyen ve gerekli güncelleme/değişiklik yapılmayan analiz ve planlamalar müşteri talepleri ile uyuşmayan veya kullanıcı ihtiyaçlarını tam olarak karşılamayan ürünler neden olur.

Agile Yöntemler



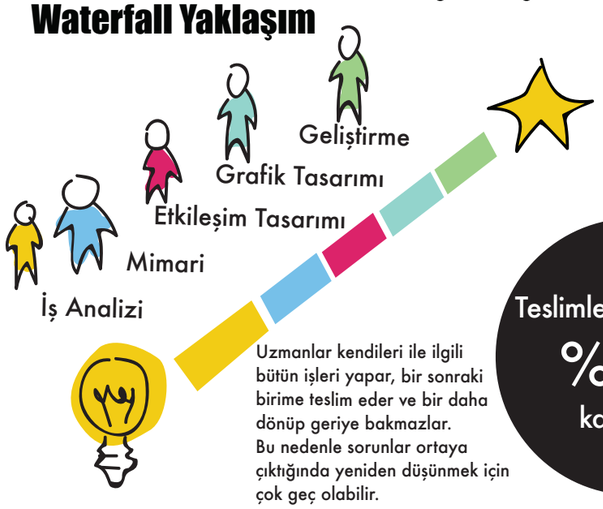


Poster: Mia Kolmodin
www.dandypeople.com
Çeviri: İlyas Varol

DANDY PEOPLE

Çalışma Yöntemleri

Waterfall Yaklaşım



Uzmanlar kendileri ile ilgili bütün işleri yapar, bir sonraki birime teslim eder ve bir daha dönüp geriye bakmazlar.
Bu nedenle sorunlar ortaya çıktığında yeniden düşünmek için çok geç olabilir.

Teslimlerde bilginin **%50**'si kaybolur

Başarmak için hata yapmaktan korkma!



Agile Yaklaşım

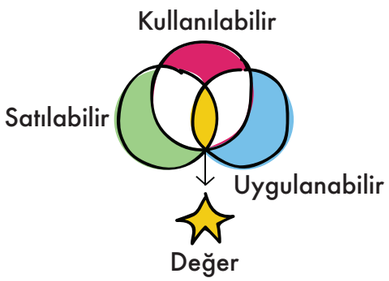
T-Şekilli Yetenekler





Çevik takımlar problemleri birlikte daha etkin çözerler.

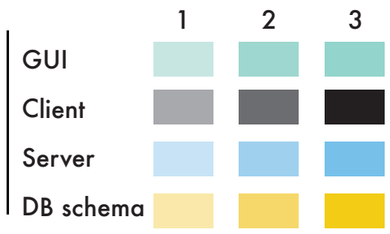
Takım Çalışması



Modern Agile



Sürekli & Sık Teslimat



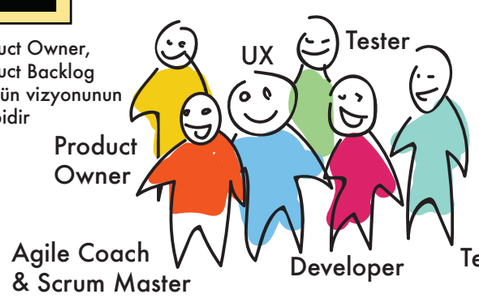
Product Backlog



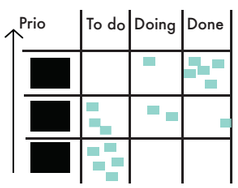
Product Owner, Product Backlog ve ürün vizyonunun sahibidir

Scrum Takımı

Yan yana çalışan çevik takımlar müşteri ile iş birliği içerisinde birlikte çözüm üretir ve değer sunarlar. Bunu yaparken karar alma yetkisine sahiptirler.



Scrum Board



- Sprint Backlog
- User Story
- Task
- DoD

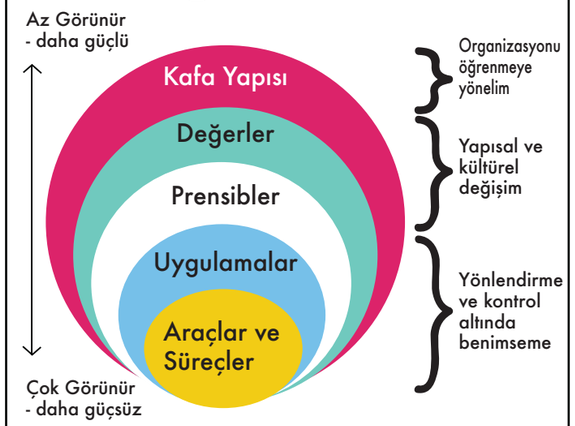
Sprintler



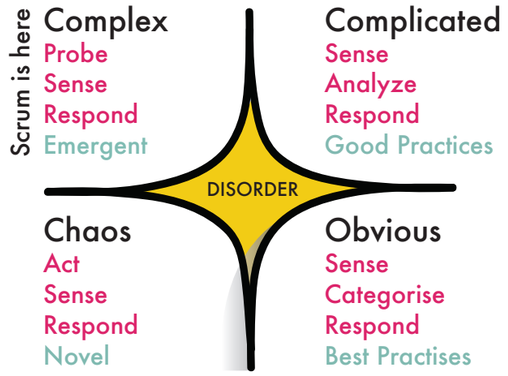
Daily Toplantıları <= 15 dk.
- Backlog Güncellemeleri / İyileştirmeleri
- Sprint Planlama
- Review
- Sprint Hedefi
- Retrospektif



Agile Olmak



Cynefin



- Cynefin'e göre bir olayın/ problemin 5 karakteristik özelliği vardır.



Büyük tasarımlar yerine İteratif tasarım



Ayrıntılı planlar yerine deneme/yanılma



Önseziler yerine Müşteri geribildirimleri



Tek bir kahraman yerine Takım Çalışması

Agile Onion AWA : Simon Powers
Cynefin : Dave Snowden
Modern Agile : Joshua Kerievsky

Agile Hikayemiz



Teşekkürler

Agile yolculuğumuzda;

- 6 yıldır dönüşümümüzün bir parçası olan ve birlikte başardığımız tüm çalışma arkadaşlarımıza,
- bizleri her zaman destekleyen, yepyeni bir konuda cevapları birlikte bulmak noktasında güçlü bir işbirliği kuran tüm yönetim ekibimize,
- tüm süreçte bize liderlik eden Tribe Lead'lerimize,
- kültürel dönüşümü mümkün kılan İnsan Kaynakları departmanımıza,
- yeni Agile çalışma şeklimizin ana unsurları olan Product Owner'larımıza, Chapter Lead'lerimize, Scrum Master'larımıza, Agile Koç'larımıza,
- desteklerini hep hissettiren ACM ailesine,

gönülden teşekkürler!

vodafone

Kitapta Katkısı Bulunanlar

Bu kitabın hazırlanmasına katkı sağlayan;

Aslı Odabaş, Ayda Yörükoğlu, Ayfer Tarım, Başak Dumanol, Barış Yücel, Berkay Murtaz,Çağla Bayrak, Çağlar Sak, Devrim Yıldırım, Didem Çalışkan, Doğan Akça, Doğu Kır, Duygu Balcı, Ekin Köktürk, Emre Doğan, Ender Özgün, Esra Erciyas, Fatih Özcan, Figen Erkılıç, Gamze Yeşilbayrak, Gülnur Bayhan, Hande Park, Jülide Özbülbül, Kıvanç Sağ, Melike Ak Şahin, Murat Büyükkucak, Murat Elmas, Murat Hacısalihoğlu, Mutlu Çiçek, Neslihan Ayyıldız, Ozan Şahin, Ömür Erdem, Özgür Tellal, Sami Dönmez, Seda Simur, Selin Sert, Serdar Şahin, Sevil Kayaş Yılmaz, Seyhan Yılmaz, Şükriye Dal, Türker Aslan, Tufan Çetiner, Tuğçe Boyacı, Uğur Sennaroğlu, Vahyeddin Tutuş, Vildan Karaca, Yaprak Çalkan

ve tüm Vodafone ailesine teşekkürlerimizle!

vodafone

Önsöz

Dijital çağın çalışma biçimi Agile



Colman Deegan
CEO
Vodafone Türkiye

Dünya değişiyor. Sektörümüz değişiyor. İş modelleri değişiyor. Bu hızlı değişim ortamında her şirket gibi bizim de değişmemiz kaçınılmazdı. Müşteri beklentilerine çok daha hızlı ve yenilikçi biçimde cevap verebilmek için organizasyon modelimizi sadeleştirmemiz ve daha çevik bir yapıya dönüşmemiz gerekiyordu. Uçtan uca müşteri memnuniyeti sağlamanın yolu, dijital çağın çalışma biçimi Agile'ı benimsemekten geçiyordu.

Agile yolculuğumuz böyle başladı. Bugün artık, dijitalleşme hedefimizle birlikte çalışma şeklimize daha çok odaklanıyor ve özellikle geliştirme süreçlerimizde Agile yaklaşımını yaygın olarak kullanıyoruz. Organizasyonel olarak hız, esneklik ve verimlilik kazanmayı hedeflediğimiz bu yaklaşımla, dijital alanlara öncelik verip bu alanlardaki hizmetlerimizi ve varlığımızı kuvvetlendiriyoruz.

Agile işbirliğini destekliyor, yalın ve hızlı olmaya, çabuk karar almaya, esnek olmaya dayanıyor. Müşteri deneyimine daha fazla odaklanmayı, pazar dinamiklerine daha hızlı yanıt vermeyi ve dijital teknolojilerden en iyi şekilde faydalanmayı mümkün kılıyor. Agile sayesinde, bir ürünün hazırlanması ve müşteriye sunulma hızında, verimlilikte, kapasite geliştirmede, lansman tarihlerinin performansında ve kalite göstergelerinde önemli artışlar kaydettik, kaydetmeye devam ediyoruz. Agile, müşteri deneyimini iyileştirmeye odaklanmamızı sağladı ve sadece bu amaca yönelik çalışan arkadaşlarla ekibimize güç kattık.

Şimdi, bu macera dolu yolculuğumuzu iş dünyasıyla paylaşma vakti; çünkü bizce, bilgi paylaşıldıkça güzel ve anlamlı... Ne yaşadık, ne hissettik, ne anladık, ne yaptık.... Hepsini bu kitapta siz değerli okurlarımız için ayrıntılı bir biçimde anlattık. Anlatmaktan büyük mutluluk duyduk.

Kuşkusuz, Agile yolculuğumuz devam ediyor. Öğrenecek ve paylaşacak daha çok şeyimiz olacak.

Tekrar görüşmek dileğiyle...

vodafone

Önsöz

Büyük bir kurum eski alışkanlıklarını bırakıp bir gün gerçekten Agile olabilir mi? Olabilirse, bunun için ne gerekir? Ya da hepsinden önce Agile olmak gerçekten nedir?

Bizim için Agile, en özet tanımıyla yenilenebilmek demek. "21. yüzyılın cahilleri öğrenmeyen, öğrendiği yanlışlardan vazgeçmeyen ve yeniden öğrenmeyenler olacak" demişti ünlü füturist Alvin Toffler. Bundan yaklaşık 50 sene öncesinde çevik yeni dünyayı bu sözlerle tanımlamıştı.

Vodafone Türkiye, 6 yıldır kendisini sürekli yenilediği Agile yolculuğunda ilerliyor. Biz de en başından bu yana bu yolculuğa eşlik ediyor olmaktan ötürü büyük mutluluk duyuyoruz. Çeviklikten gelen deneme ve yeniden öğrenme süreci Vodafone Türkiye için önemli odak noktalarından birisi. Bu kültürün şirketin DNA'sı haline gelmesi hedefiyle çalışmalarımıza devam ediyoruz. Geride bıraktığımız 6 yıllık süreçte epey yol aldığımızı görmek umut verici.

İşte bu kitapla birlikte bugüne kadar bu yenilenme yolculuğundaki kazanımlarımızı, hatalarımızı ve öğretilerimizi sizlere sunmak ve hep birlikte paylaşarak öğrenmeyi tetiklemek istedik. Yenilenme sürecini anlatmayı hedeflediğimiz kitabımızı da içeriğin ruhuna uygun şekilde kendini yenileyebilen, Agile bir kitap olarak tasarladık. Okuyacağınız kitap çevik dönüşüm yolculuğumuzun ilk çıktısı. Bu ilk çıktı ile Vodafone Türkiye çalışanlarının gerçek hayat deneyimlerini ve duygularını sizlerle paylaşmayı amaçladık.

Kitabımızın kendi serüvenine yeni içerikler ekleyerek bir fidan gibi büyümeye devam etmesini hayal ediyoruz. Bu içerikleri de sizlerden gelecek yorumlarla şekillendirmeye çalışacağız ve arayı çok uzatmadan yeni bölümler ekleyerek kitabı güncelleyeceğiz.

İleriki bölümlerde tekrar görüşmek ve gelecekte sizlerin de ilham verici dönüşüm yolculuklarını okuyabilmek dileğiyle...



Mehmet Yitmen
Yönetici Ortak
ACM

vodafone

Vodafone, Benim Çevik Hikayem

İçindekiler

Bölüm 1: Değişen Yapımız ve Roller

Neden Agile?
Yeni Agile Yapımız
Şu Ana Kadarki Öğretilerimiz
Gelecekte Bizi Bekleyenler
Squad Nedir?
Product Owner Kimdir?
Scrum Master Kimdir?
Chapter Lead Kimdir?
Tribe Lead Kimdir?
Agile Koç Kimdir?

Bölüm 2: Henüz karar vermedik. Siz okuyucularımızdan gelen yorum ve istekler yönünde şekillenecektir.

vodafone

IT içerisindeki ilk pilot
Agile takım

Pilot takım sayısının 6'ya
çıkarılması

IT içerisinde Agile Solutions
birimin kurulması

Teknoloji altında Network ve
Operasyon ekiplerine yayılım

Vodafone Grubu şirketleri arasında
Agile mükemmeliyet merkezi seçildik ve
hizmetlerimize başladık

Agile'ın teknoloji ekipleri dışında yayılımı
(Örn. İnsan Kaynaklarındaki ilk uygulamalar)

02 2018 İlk Tribe Yapısı:
Digital Accelerator Tribe

09 2018 Digital Assets Tribe Kurulumu

01 2019 Digital Experience Engineering
Tribe Kurulumu

02 2019 Core IT Squad'larının
Kurulumu

03 2019 CBU Growth Tribe
Kurulumu

04 2019 CBU Acquisition Tribe
Kurulumu

07 2019 Care Tribe
Kurulumu

Genel Bakış: Vodafone Türkiye'nin Agile Yolculuğu

Vodafone Türkiye'nin Agile Yolculuğu

Birinci aşama: Pilot Çalışması (2013 – 2014)

- 2013 yılının Aralık ayında ilk pilot çalışmamızı IT ekibimizde yaptık.
- Bu çalışma için 6 kişilik bir Agile ekibi kurduk.
- Bu çalışmada en çok odaklanılan alanlar takım çalışması ve şeffaflık olmakla birlikte 3 ay süren çalışma sonunda, proje kapsamındaki müşteri taleplerini karşılama hızı 3 kat arttı. (120 günde üretilen bir iş 40 günde üretilmeye başlandı.)

İkinci aşama: Yaygınlaştırma (2014 – 2018)

- 2014 Şubat ayında pilot projede elde edilen başarıyı yaygınlaştırmak için yeni Agile IT ekipleri kurduk.
- Agile yöntemini başka ürün ve servisler için de denedik.
- Agile yöntemini yaygınlaştırdığımız 2 yıl içinde:
  - IT'de Agile çözümlerine dedike bir organizasyon birimi kurduk. Bu birim, zaman içinde yeni çözümleri de sahiplenerek büyüdü.
  - IT'ye gelen büyük çaplı olmayan tüm talepler için Agile yöntemini uyguladık. 2016 yılında IT projelerinin yaklaşık %75'i Agile yöntemi ile gerçekleştirildi.
  - Ticari ekiplerin de Agile çalışma yöntemini tanıyarak daha çok projeye dahil olmalarını sağladık.
  - Agile yönteminin IT dışındaki teknoloji birimlerinde (Network Operasyon, Network Planlama, vb.) yaygınlaşmasını sağladık.
  - 200'ü aşkın kişi ve 30'u aşkın Agile takımı ile bu süreci tamamladık.
- Tüm bu inisiyatiflerimiz sonucunda Vodafone Türkiye olarak Vodafone Grubu şirketleri arasında mükemmellik merkezi seçildik ve diğer ülkelerdeki Vodafone Grubu şirketlerine koçluk hizmeti vermeye başladık.
- Yaygınlaştırma aşamasında kazanımlarımız neler oldu diye bakarsak:
  - IT ve ticari fonksiyonların iş birliği gelişti.
  - Pazara çıkış süresi kısaldı, ürün kalitesi ve üretim arttı.
  - Takım ruhu gelişti ve şirket içinde takım çalışmaları yaygınlaştı.
  - Yeni neslin çalışma şekline uyumlu bir şekilde çalışılan alanlarda hızlı sonuçlar alındı.
- Yaygınlaştırma aşaması tamamlandıktan sonra, ticari fonksiyonların Agile çalışma yöntemini değerlendirmeleri için bir anket gerçekleştirdik. Anket sonucunda ticari fonksiyonların toplam memnuniyet skorunun 5 üzerinden 4,2 olarak ölçümlenmesi, yaygınlaştırma aşamasının başarısını gösteren başka bir faktör oldu.

Üçüncü aşama: Agile Çalışmanın Benimsenmesi ve Kültürel Dönüşüm (2016 – devam ediyor)

- 2016 Kasım ayından günümüze kadar olan sürede kültürel dönüşümü sağlamak için birçok inisiyatif başlattık.
- Bu süreçte en önemli adımlardan biri Agile yöntemini şirketin teknoloji harici fonksiyonlarında kullanmaya başlamamız oldu.
- Şirket genelinde çalışanlara Agile eğitimi verdik.
- Şirkete Agile'ı tanıtmak ve kazanımlarından bahsetmek için kısa oturumlar düzenledik.
- Bu süreçte yeni bir konsept olan Tasarım Odaklı Düşünce Şekli (Design Thinking) ile tanıştık.
    - Yeni fikirler üretme ve geliştirme sürecinde bu uygulamadan faydalanmak amacıyla şirket çalışanlarının katılımıyla atölye çalışmaları düzenledik.
    - Agile çalışma yönetimini destekleyen bu uygulama ile müşterinin ihtiyacı olan doğru inovatif servis, süreç ve ürünlerin tespit edilmesini amaçladık.
- 2018 itibarıyla Agile çalışma yöntemini ticari fonksiyonlar bünyesinde dijital alanlara odaklanarak kullandık. İlk Tribe (Kabile) yapımızı kurduk.
    - Dijital müşteri değer yönetimi, dijital pazarlama ve dijital satış konularında çalışan Agile ekipler kurduk.
    - Tahmin ederek değil, veriyi analiz ederek; müşterimizin ihtiyacını karşılayacak işler üretmeye öncelik verdik.
    - Bireysel iş birimimizde başlattığımız bu "Agile" yaklaşımla, ilk dönemde üretim hızında 3 kat, verimlilikte 2 kat, kapasite geliştirmede 2,5 kat, lansman tarihlerinin performansında 3 kat, kalite göstergelerinde 5 kat artış kaydettik.
- 2019'da ise kapsamı genişleterek Bireysel Pazarlama'nın tamamında dönüşüme gittik ve neredeyse tüm Bireysel Pazarlama Agile olarak çalışmaya başladı. Dijital kanal yönetimi, yeni ürün geliştirme, müşteri baz yönetimi ve teknoloji konularında farklı fonksiyonlardan kişilerin bir araya geldiği Agile ekipler kurduk.
- Agile pratiklerini organizasyon seviyesinde daha çok çalışana ve departmana ulaştırabilmek için 2019 başında İç Agile Koç Yetiştirme Programı başlatıldı. Vodafone tarihinde ilk kez, pozisyona özel olarak tasarlanmış değerlendirme ve seçme sürecinden geçen adaylar ile Agile Koç takımı oluşturuldu.

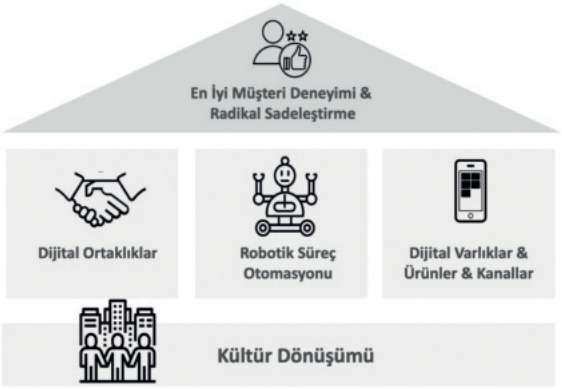
Bölüm 1: Değişen Yapımız ve Roller

Neden Agile?

Sürdürülebilirlik, günümüzde şirketlerin Agile dönüşüm çalışmalarına girmelerindeki en büyük motivasyon olarak öne çıkıyor. Şirketlerin hayat süreleri kısaldı. Geçtiğimiz on senede en büyük şirketler tablosu neredeyse tamamen değişti. Milyarlarca dolar değerindeki global devlerin hızlıca yok olduğuna şahitlik edebiliyoruz. Bu kadar demokratik bir pazarda, iki üç kişinin bir garajın içinde rakip olabileceği bir ortamda şirketi sürdürebilecek misin? Bu kadar çok şirketin böyle büyük bir dönüşüme cesurca girmesinin ardında bu zorlu ortam koşulları yatıyor. Daha hızlı ve çevik olarak değişen pazar şartlarına ayak uydurmak istiyorlar. Organizasyonlar radikal bir şekilde değişiyor, dikey yapılar ortadan kalkıp yatay yapılara geçiyor.

Biz de Vodafone Türkiye olarak "Biz Dijitaliz, Biz Geleceğiz" mottosuyla kapsamlı bir dönüşüm programı başlattık. Dijital ürünler, robotik süreçler, dijital iş ortaklıkları ve kültürel dönüşüm başlıklarından oluşan bu programla hem çalışanlarımız hem de müşterilerimiz nezdinde daha çevik, daha hızlı, daha sade ve dijital geleceğe hazır bir şirket olmayı hedefliyoruz. Kültürel dönüşüm başlığı altında odaklandığımız en önemli konu ise Agile çalışma biçimine ağırlık vermek oldu. Organizasyonel olarak hız, esneklik ve üretkenlik kazanmayı hedeflediğimiz bu yaklaşımla, dijital alanlara öncelik verip bu alandaki hizmetlerimizi ve varlığımızı kuvvetlendirmek istiyoruz.

Şirketimizde inşa ettiğimiz Agile yapı ile şu anda farklı uzmanlıklara sahip 8-12 kişilik ekipler hiyerarşik olmayan bir düzen içinde aynı amaç için bir arada çalışıyorlar. Otonom yapıları ile ürün ve süreçlerini sürekli iyileştirerek, iş



hedeflerine ulaşmak için çalışan bu takımlar, yazılım geliştirme uzmanından marka müdürüne kadar farklı profiller barındırıyor. Takımı oluşturan bireyler, kendi uzmanlıklarıyla ilgili sorumluluk almanın yanında, takımca odaklandıkları hedeflere ulaşmak için farklı uzmanlıklardan da sorumluluk almaya gayret ediyorlar. "Squad" adı verilen bu ekipler, kendi varoluş amaçlarına yönelik olarak uçtan uca sorumluluğa; yani hem yetkiye hem her türlü donanıma hem de bütçeye sahipler. Bürokrasi mümkün mertebe ortadan kalkmış olduğu için kararlarını hızlıca alabiliyor ve hayata geçirebiliyorlar. Ast-üst ilişkisi yerine kendi kendine yeten ve kendi kendini yönetebilen takımlar olduklarından, olasılıkları hep beraber konuşup en iyi karara ve çözüme ulaşmaya çalışıyorlar. "Kısa dönemlerle, müşteriye değer katan küçük çıktılar üret, böylece hata varsa çabuk gör ve çabuk düzelt" mantığıyla çağın gerektirdiği hızda ve çeviklikte hareket ediyorlar. Çözüm odaklı olmayı sağlayan ve bireyleri motive eden Agile çalışma şekli, ekip arkadaşlarımızın daha fazla inisiyatif almasına izin veriyor ve gelişimlerini destekliyor.

Bu anlamda Agile çalışma şeklinin çok faydasını gördük, görmeye devam ediyoruz. Bazı noktaları ön plana çıkarmak gerekirse;

- Agile, iş birliğini destekliyor. Bir hizmet geliştirme sürecinin gerekli tüm fonksiyonlarını bir araya getiriyor, siloları ortadan kaldırırken takımlara esneklik ve hız kazandırıyor.

- Yalın ve hızlı olmanın, çabuk karar almanın yanı sıra esnekliğe dayanan bu sistem; müşterilerimizin gerçek ihtiyaçlarına odaklı, katma değer yaratan dijital ürün ve servisler geliştirilmesine imkân veriyor.

- Müşteri deneyimine daha fazla odaklanmayı, pazar dinamiklerine daha hızlı yanıt vermeyi ve dijital teknolojilerden en iyi şekilde faydalanmayı mümkün kılıyor.

- Agile çok şeffaf bir yapı. Hedeflerin konulması, sonuçların takibi, işlerin yapılışı ve iş yükünün paylaşımı çok şeffaflaşıyor. Departmanlar arası "sen yapıyorsun, ben yapmıyorum, ben yapıyorum, sen yapmıyorsun" gibi çekişmeler tamamen ortadan kalkıyor. Herkes o işin neden yapıldığını, neden yapılamadığını ve yapılamıyorsa eksiğin nerede olduğunu çok net görüyor.

- Yeni roller ve ritüeller, odaklanma ile birlikte disiplin getiriyor ve üretkenliği tetikliyor.

Bahsetmiş olduğumuz iyileştirmeler rakamlarımıza da yansıyor. Bireysel iş birimimizde başlattığımız bu "Agile" yaklaşımla, ilk dönemde üretim hızında 3 kat, verimlilikte 2 kat, kapasite geliştirmede 2,5 kat, lansman tarihlerinin performansında 3 kat, kalite göstergelerinde 5 kat artış kaydettik.

Hızlanmaktan bahsetmişken, işin bu kısmını biraz açmak gerekiyor. Yeni yapıyla birlikte hızlanmak için önce yavaşladık ve sonra hizalanarak üretkenliğimizi artırdık. Önceki yapımızda pazarlama ekipleri gecelere kadar çalışıyor ve çok sayıda proje üretiyordu. Ancak, çoğu projenin hayata geçmesi IT'ye bağlı. IT'ye, birçok farklı departmandan fikirler, projeler yağmur gibi yağıyor; kapasitesi belli olan IT de ister istemez darboğaz konumunda olduğu için, bu çok sayıdaki proje büyük güçlükle ilerlemeye çalışıyordu.

Peki öyleyse bu kadar çok fikir üretmeye ve hepsini hayata geçirmeye çalışmaya gerek var mıydı? Hayata geçiremediğimiz fikir sayısı arttıkça aslında sistemin üzerindeki tansiyonu da artırdığımız için iyi bir şey yapmadığımızı fark ettik. Dolayısıyla şu an çevikleşip ekipleri birleştirdik ve aslında üretim hızımızı da geliştirme hızına göre ayarlamaya başladık. Başka bir deyişle organizasyonu bilerek yavaşlattık. Ancak, elbette, bu yavaşlama iş çıktımızın azaldığı anlamına gelmiyor. Aksine üretkenliğimiz, toplu üretimimiz arttı. Organizasyonu hizalayarak ve senkronizasyon sağlayarak aynı süre içinde en doğru değeri üretmeye odaklanıyoruz. Böylelikle çok büyük çoğunluğun fazla mesai yapmak zorunda kalmadığı ve daha mutlu olduğu bir organizasyona sahip olduk. Sonuçlarımız da daha iyi.

"Hepimiz bir evrimden geçiyoruz. Ben ilk iş hayatıma başladığımda sekizde çıkmak ayıptı. Yerleşmiş bir alışkanlıktı geç saatlere kadar kalmak. Hatta insanlar erken çıktıklarında kendilerini kötü hissediyorlardı. Ama hepimiz değişiyoruz. Şu an bakıyorum resim bu, artık beşte herkes rahatça çıkıyor. Bunun yanı sıra herkes istediği yerde ister evinde ister kafede çalışıyor. Düşününce tüm bunlar büyük değişimler. Böyle bir ortamda eskisi gibi kişileri takip etmek de mümkün değil. Zaten önemli olan kişilerin ne yaptığı da değil. Tek baktığımız şey toplamda işin çıktısı. Çıktı varsa yani iş değeri, doğru yoldayız. Çıktı yoksa, sabah beşe kadar hep birlikte kalsak ne fark eder. Hatta daha kötü bir şey."

Yeni Agile Yapımız

Bugün Core IT[1]'deki Agile yapılanmamız ve bireysel iş birimimizdeki 5 Tribe'ımız ile organizasyonumuzda müşteri deneyimi odaklı bir işleyiş oluşturmak için Agile yapıya geçmiş bulunuyoruz. Bu Tribe'larımız;

- **Acquisition Tribe (Yeni Müşteri Kazanımı):** Yeni müşteri kazanımından sorumlu; müşterilerimize en etkileşimli dijital ürünleri sunuyor.
- **Base Growth Tribe (Baz Müşterinin Büyümesi):** Daha fazla müşteri etkileşimi ile bazımızın artırılmasından ve değerinden sorumlu.
- **Digital Assets Tribe (Dijital Varlıklar):** Müşterilerimize en iyi dijital kanal deneyimini yaratıyor.
- **Care Tribe (Müşteri Hizmetleri):** Müşterilerimize özelleşmiş ve veriye dayalı teknolojilerle müşteri hizmetleri sunuyor.
- **Digital Experience Engineering Tribe (Dijital Yetkinlik):** Müşteri deneyimini iyileştirmek için yeni kabiliyetler geliştiriyor.

Tribe'ı tanımlayarak başlarsak; bir amaç etrafında toplanan, ortak değerlere sahip, belirli bir düzen içinde işleyen topluluktan bahsediyoruz. Bu tanımda amaç kelimesinin altını çizmek gerekir. Bizim için amaç, o ekibin varoluş sebebi demek. Bir örnek verecek olursak; Acquisition Tribe, müşterilerimizin çok sevecekleri, hatta âşık olacakları, dijital deneyime sahip ürünler yaratmak için var. Bu Tribe'ın içindeki her Squad da bu amaç etrafında kurgulandı. Mesela, Acquisition Tribe'ındaki bir Squad faturalı müşteri deneyimini mükemmelleştirmeye çalışırken, bir diğeri akıllı telefon deneyimi üzerine odaklanıyor.

Bireysel iş birimimizdeki bir önceki yapılanmamız fonksiyon bazlı klasik bir yapıydı. Bu yapıyı müşteri deneyimi ve yolculuğunu göz önünde bulundurarak Tribe'lar halinde şekillendirdik. En başta ürün ya da segment odaklı alternatif yapıları da değerlendirdik. Müşteri deneyimindeki adımları izleyen yapıyı kurmamızın birkaç sebebi var. Birincisi, müşteri yolculuğunun her adımında uzmanlaşmış ve bu adımları her gün daha mükemmel hale getiren ekipler yaratmak istedik. Bunu yaparken de diğer adımlara odaklanan ekiplerle uyum içinde çalışmalarını hedefledik. İkincisi, müşteri yolculuğundaki her safhanın farklı dinamikleri olduğu gerçeğiyle hareket ettik. Acquisition Tribe, müşterileri Vodafone'a kazandırmaya çalışıyor. Pazar dinamiklerini yakından takip edip, en çekici kampanyaları ve ürün deneyimlerini tasarlıyorlar. Bir diğer Tribe, Vodafone'un mevcut müşterilerinin hayat döngüsünden sorumlu. Müşterilerimizden ulaşan verileri inceleyip, her abonenin farklılık gösteren ihtiyaçlarına göre ürün ve servisleri tam da ihtiyaç duydukları anda onlara ulaştırmak için uğraşıyorlar. Son olarak her Tribe farklı dijital teknolojileri kullanıyor. Müşteri çekmek için performans pazarlama teknolojilerinden faydalanırken, abonelerimizin sadakatini artırmak için analitik modellemeler yapıyoruz.

Sonuç olarak tüm faktörleri değerlendirince, abonelerimizin bizimle yaşadıkları yolculuğu

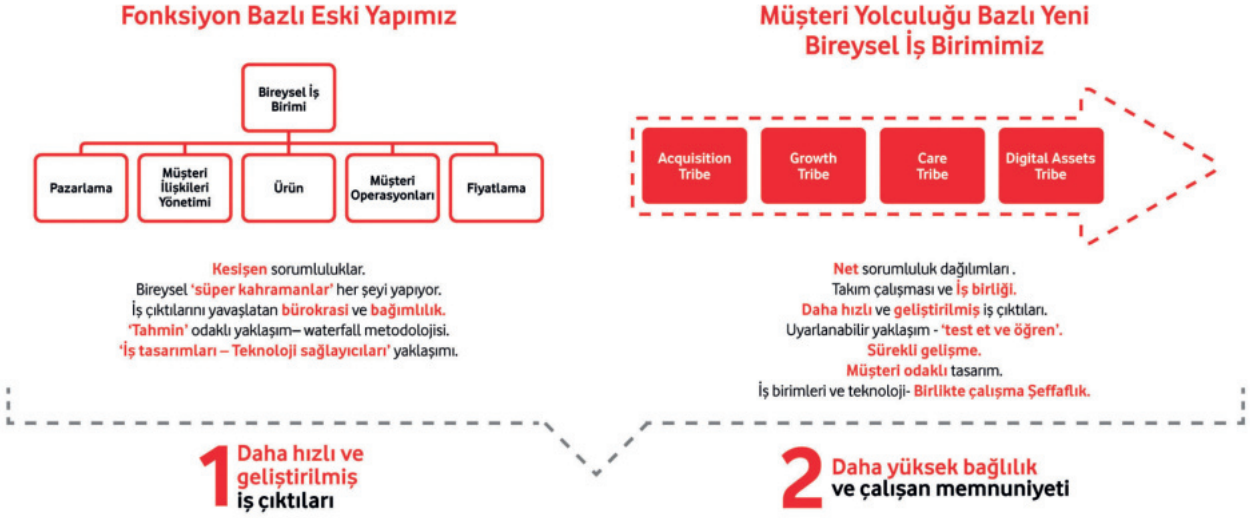
[1]**Core IT:** Vodafone Türkiye organizasyonu içerisinde Tribe'ların dışarısında kalan IT ekiplerinin oluşturduğu yapıya verdiğimiz isim.

ve bu yolculukta yaşadıkları deneyimlerden öğrendiklerimizi kullanmaya karar verdik.

Yeni yapımızı başlangıçta İcra Kurulu Üyelerimizin, İnsan Kaynakları'nın, Tribe Lead'lerin ve Agile konusunda uzman danışmanların oluşturduğu bir grup ile çalıştık. İlk kurguyu tasarlamak bu ekibin göreviydi. Her şey doğru oldu mu diye sorarsak, olmadı tabii ki. Doğru olması da gerekmiyordu. Bu yapının da değişen ihtiyaçlar ve pazar koşulları doğrultusunda zamanla değişmesi ve evrimleşmesi gerekiyor.

Tribe'ları müşteri yolculuğuna göre tasarladıktan sonra sırayı Squad'lar aldı. Her Tribe Lead, yukarıda bahsettiğimiz genel yapıyı tasarlayan ekibin de desteğiyle birlikte kendi iç detaylarını belirledi. Squad'lar birkaç temel unsur doğrultusunda oluşturuldu. Birincisi, kurulma amacına ulaşmayı sağlayacak gerekli tüm uzmanlık ve yetenekleri Squad'a dahil etmiş olmaktı. İkincisi ise tüm bireylerin bu amacı net olarak anlamasını sağlayabilmekti. Bu farkındalığın, otomatik olarak sorumluluğu da beraberinde getireceği inancı hakimdi. Tribe'ın farklı Squad'lardan oluştuğundan hareketle, Squad'lar arası sınırların, gri alanların da belirlenmesi, uyulması gereken genel politikaların farkında olunması ve başları sıkıştığında nereden destek alabileceklerinin iyi anlatılması gerekiyordu.

Şu anda Core IT ve diğer Tribe'larımızda 50'den fazla Squad'ımızla birlikte çalışıyoruz. Her Squad'ın çok net hedefleri var ve bu hedeflere ulaşmak için takımların ihtiyaç duyduğu tüm yetkinlikleri takıma dahil etmeye çalışıyoruz[2]. Ticari yetkinlik, pazarlama, ürün, satış, test, yazılım geliştirme gibi yetkinliklerden hangisi takım için gerekli ise, o yetkinlikteki arkadaşlar takımın içinde yer alıyor. Bu yapının doğru ve en verimli şekilde işlemesi için de takımlara Agile Koç'larımız destek veriyor.



[2]**İşe Uyum Kazandırma**
Kişilerin yeni çalışma şekline adaptasyonunu sadece eğitim yolu ile sağlamak yeterli değil. Yönetim ekibinden, Tribe Lead'lere, Chapter Lead'lerden Scrum Master'lara kadar birçok kişi için yeni rollerine uyumlamaya yönelik aksiyonlar aldık. Bu rollere yönelik özel yetkinlik gelişim programları oluşturduk. Rollerle yaptığımız birebirler sonucunda gelişim yönlerini belirledik. Ekip arkadaşlarımızı hem zihnen hem de kültür olarak olması gereken noktaya getirmek hala odağımızdaki bir konu.

Uzmanlıkların Desteklenmesi

Agile Dönüşümün önemli parçalarından biri de uzmanlık. Özellikle Squad'ları oluştururken o Squad'ın ihtiyacı olan yetkinlikleri sağlamak önemli. Aksi takdirde otonomiyi ve akabinde başarıyı yakalamak çok zor. Biz bu aşamada yola insanlardan başlayarak değil, daha ziyade Squad'ın hedefleri doğrultusunda gereken yetkinliklere odaklanarak çıktık. Bu yetkinliklerin bir kısmı halihazırda bizim organizasyonumuzda olan yetkinliklerdi; temel pazarlama, yazılım geliştirme gibi örnekler sayılabilir. Dolayısıyla bu yetkinliklere sahip çalışanlarımızı rollerle eşleştirmek çok da zor olmadı. Fakat diğer yandan UX (Kullanıcı Deneyimi), DevOps gibi organizasyonumuzda olmayan veya göreceli olarak az bulunan bazı yetkinliklere de ihtiyaç olduğu açıktı. İşte bunlar için ciddi bir işe alım süreci yaşadık. Maalesef bu yetkinlikler genel olarak da kolay bulunmuyor. Dolayısıyla iki soru çok ön plana çıkıyor. Bunlardan bir tanesi "az bulunan bu yeteneklere farklı yollarla nasıl ulaşırım?" Diğer soru ise "bulduğum bu kişilerin gerçekten ilgili uzmanlığa sahip olup olmadığını nasıl anlarım?" İşte bu iki soru bizi işveren markası ve işe alım süreçlerinde farklı yaklaşımlara götürdü.

Örneğin, Mayıs 2019'da bir Designathon yarışması yaparak Türkiye'nin farklı yerlerinden servis, ürün ve deneyim tasarım alanında deneyimli kişilerle beraber olduk. Bu tarz yarışmaların hem aday çekme hem de aday kaynağı yaratma anlamında etkili olduğunu düşünüyoruz. Staj programlarımızı da ihtiyacımız olan yetkinlikler ve rollere göre yapılandırdık. Teknoloji ve Satış alanlarında kadın çalışan oranımızı artırmak amacıyla yürüttüğümüz Teknolojide Kadın ve Satış Fonksiyonunda Kadın Staj Programlarımıza geçen yıl "Geleceğin Meslekleri Staj Programı"nı ekledik. Bu programla alınan tüm stajyerlerimizi dijital rollerde ve Squad'larda konumlayarak mezuniyetleri sonrasında onları Discover Genç Yetenek Programı'mıza dahil ettik. 2019 yılında bu programlarla staj yapan öğrencilerin %65'i yeni mezun programımızla kariyerlerine Vodafone'da devam etme kararı aldı.

Bunları yaparken bir yandan işe alım süreçlerimizde de yaklaşımımızı değiştirdik. Yaptığımız Değerlendirme Merkezi süreçlerinde sosyal beceri ölçümlerinin yanında ilgili alandaki uzmanlık seviyesini de ölçümleyecek teknik vakaları da kullanmaya başladık.

Genel İşleyişimiz

Her şeyden önce her Squad çalıştığı ürüne ya da deneyimine göre kendi Agile süreçlerini ve kendi ritmini belirliyor. Bütün yapımız bu temel üzerinde kuruldu. Squad seviyesinden baktığımızda, Agile çalışma yönteminin olmazsa olmazı olan tüm ritüelleri aksatmadan uygulamaya çalışıyoruz ve bu yapıyı hem Scrum Master'larla hem de Agile Koç'larla destekliyoruz.

Squad'lar içindeki roller için topluluklarımızı oluşturmaya başladık. Scrum Master, Product Owner, Veri Uzmanı, vb. topluluklarımız var. Bu oluşumlarla, arkadaşların benzer alanda çalışanlarla bilgi paylaşımı yapmalarını ve iyi uygulamaların yayılmasını amaçlıyoruz. Öğrenmeyi ve gelişimi tetiklemek adına bu toplulukları çok önemsiyoruz.

Öğrenme ve gelişim ile ilgili olarak diğer bir yapımız da yetkinlik bazlı yatay oluşumlar olan Chapter'larımız. Chapter'larda ilgili alandaki iş yapış tarzını ve şirket içindeki bu kabiliyetin nasıl geliştirileceğini tartışıyoruz ve destekliyoruz. Chapter Lead'ler bu yapı içinde uzmanı oldukları alandaki bireylerin yetkinlik gelişimini sağlıyor ve yetkin, güçlü, otonom bir yapıyı destekliyorlar.

Tribe seviyesinde şirket stratejilerine hizalanmak, ana iş hedeflerini belirlemek Tribe Lead'in görevi. Nasıl ki Vodafone Türkiye, globalde Vodafone Grubu'nun bir parçası ve grubun hedefine ulaşması için Türkiye'nin de kendi hedeflerini gerçekleştirmesi gerekiyorsa; aynı ilişki Tribe ve ana şirket özelinde de mevcut. Tribe'lar Vodafone Türkiye'nin lokal parçası gibi düşünülebilir. Şirket stratejimiz net, hedefleri de belli olduğuna göre, sonraki adım olarak bu hedeflerin elde edilmesi için hangi Tribe'ın ne yapması gerektiğinin belirlenmesi gerekiyor. Bu noktada asıl konu Tribe'in stratejisinin ve vizyonunun ana strateji ve vizyon ile uyumlu olması. Tribe'ın vizyon ve stratejisi yukarıdan gelmiyor. CEO ya da yardımcılarının görevi ana stratejiyi çizmek iken, Tribe Lead'in görevi ise kendi Tribe'ının stratejisini bu ana stratejiye uyumlandırmak. Biz yine bunu da Tribe'larımızın ilk kuruluşunda yaptık. Bu stratejilerin iletişimi ve ekiplerin bu hedefleri içselleştirmesi çok önem verdiğimiz bir konu. Tüm Tribe içinde benimsenmesi için de birkaç defa hep birlikte üzerinden geçtik. Bu hedefler, Tribe Lead ve Product Owner'ların hizalanması ve sürekli olarak tüm ekip iletişiminin sağlanması ile birlikte Squad hedefleri olarak sahipleniliyor. Squad'lar hedefleri bu şekilde içselleştirdikleri için de kendi üretim ritimlerinde hedeflere doğru koşmaya başlıyorlar.

Bunun yanında Tribe içindeki teknoloji ile ilgili her türlü stratejik konu ve hizalanma Co-Lead adını verdiğimiz teknoloji liderinin sorumluluğunda. Co-Lead'in temel rolü, stratejik olarak teknik bakış açısını Tribe'a sağlamak. Teknolojik anlamdaki ihtiyaç noktasında Tribe'a biraz destek olup yön göstermesi gerekebiliyor. Bu rol tabii şu anki durumumuzda bir ihtiyaç. İleride Tribe Lead'ler teknoloji tarafına da yavaş yavaş hâkim olmaya başladıkça belki de Co-Lead ihtiyacı da kalmayacaktır. Ayrıca, Tribe'ın temel fonksiyonu olmayan finansal ve idari bazı konular da Tribe dışından atanan temsilciler aracılığı ile yönetiliyor.

Tribe'lar içinde o Tribe'ı ilgilendiren organizasyonel kararları almak için iki haftada bir yapılan Tribe Governance toplantısı var. Bu toplantıya Chapter Lead'ler, Product Owner'lar, Agile Koç ve Tribe Lead katılıyor. Ayrıca, ayda bir kez hedeflerimizi ve çalışmalarımızı Tribe içinde hep birlikte değerlendirdiğimiz Tribe Day etkinliğimiz var. Bu etkinlikle birlikte Squad'larımıza büyük resimde sahip olduğumuz takım ruhunu hissettirmeye çalışıyoruz.

Stratejik anlamda daha genel resme baktığımızda, Tribe'ların hizalanması için Tribe Lead'lerin bir araya geldiği haftalık toplantılarımız var. Üç ayda bir de şirketteki tüm Product Owner'lar bir araya geliyor ve QBR (Quarterly Business Review)

ismini verdiğimiz hizalanma etkinliğini yapıyoruz. QBR'larda Squad'lar önlerindeki üç ayın hedef ve planlarını, iş fikirlerini bir havuzun içine döküyorlar ve böylece büyük resmi herkes görüyor. İş öncelikleri ve bağımlılıklar böylelikle belirleniyor.

"Buradaki en önemli konu genel şirket stratejisi ile Tribe stratejisi, sonrasında da Squad hedefleri arasındaki ilişkiyi kurmak. Bence yaşanan en önemli zorluk, takımın otonomisi ile genel stratejinin koyduğu hedeflerin getirdiği baskı. Burada hedef baskısının iyi ayarlanması lazım ki takım deneyselliğe, sürekli iyileşme ve gelişmeye de vakit ayırabilsin. Bu esasen gerek sürekli iyileşmeye yol açması gerekse yenilikçiliği teşvik etmesi nedeniyle Agile çalışma felsefesinin temellerinden biri olarak tanımlanabilir. Bunun üst yönetim tarafından desteklenmesi çok önemli ki takımlara nefes alabilecek hava boşlukları kalsın. Yoksa oldukça disiplin ve konsantrasyon isteyen bir rutin içerisinde hedef baskısı her şeyin önüne geçer. Bu durum kısa vadede fayda sağlamakla birlikte, uzun vadede takımın enerjisinin ve motivasyonunun düşmesine neden olabilir."

Her ne kadar Agile ile birlikte birbirinden bağımsız çalışabilen bir yapıyı hayal etsek de henüz o noktada değiliz. Gerek aynı Tribe'ın içindeki gerekse Tribe'ın dışındaki yapılarla bağımlılıklar olabiliyor. Bu tür bağımlılığı olan işleri genelde Product Owner'lar kendi aralarında konuşarak yönetiyorlar. Uygulanan bir iyi pratik olarak da Sprint Review etkinliklerine o Squad'dan destek bekleyen tüm paydaşlar davet ediliyor ve önceliklendirme sürecinde kendi beklentilerini dile getiriyorlar.

Aslında özetlersek, yüksek iletişim ve iş birlikteliği hizalanmayı kolaylaştıran unsurlar. Ancak, henüz bağımlılıklar bizim için de bir engel oluşturmaya devam ediyor. Bu bizim de öğrenerek ve deneyip iyileştirerek ilerlettiğimiz bir alan.

Şu ana kadar bireysel iş biriminin kabaca dörtte üçünü Agile yapıya geçirdik diyebiliriz. Agile takımların dışında tuttuğumuz fiyatlandırma gibi bir konumuz var mesela. Fiyat bütün Tribe'ları kesen bir konu. O yüzden Tribe'lar arası koordinasyonu sağlayacak özel bir takım tarafından yönetiliyor. Bu yapılara CoE (Center of Excellence) diyoruz. CoE'ler Squad'ların içinde dağıtılamamış belirli bir uzmanlığa odaklanmış, bu uzmanlık alanında tüm diğer Squad'lara hizmet veren yapılar. CoE riskin yüksek olduğu fiyatlama ya da henüz yeterli uzmanın olmadığı Kullanıcı Deneyimi ya da stratejik olarak tek elden ilerletilmesinin daha etkin olduğunu düşündüğümüz Marka Yönetimi gibi konular için kurguladığımız bir çözüm.

Şu Ana Kadar ki Öğretilerimiz

- Agile IT dönüşümü demek değildir. Agile topyekûn şirketin dönüşümüdür. Agile kavramına sadece IT'nin işi ya da IT'nin kendi işlerini yönetimi şeklinde bakmak çok kısıtlı bir yorum olur. Agile yapı işin dönüşümü, iş yapış tarzının baştan yaratılması demek.

- Bu, geri dönüşü olmayan bir süreç. Agile'ın geri dönüşü yok! Başladın mı sonuna kadar gitmek zorundasın. Takımlara o gücü ve özgürlüğü verdikten sonra, takımlar da bunu aldıysa, geri dönmek çok yıkıcı olur.

- Agile yapının kalbi odaklanma ve izolasyon. Hem Tribe'a hem de Squad'lara net amaçlar belirleyip dedikasyonu sağlamanız gerekiyor. Nasıl ki Tribe'lar gelir üretmekten ve artışından sorumlu, yapının içindeki her Squad da örneğin "ben müşteri kaybını engellemekten ya da mevcut müşterilere yeni ürünler satmaktan sorumluyum" diyebilmeli. Bu, dedikasyon tarafı. Bir de izolasyon bacağı var. İzolasyon, takımları bağımlılıktan kurtarmak. Takımın kendi hedefine kendi kendine ilerleyebileceği ortamı sağlayabilmek. Bu, zorlu bir konu; sürekli iyileştirerek, adım adım ilerlemeniz gerekiyor.

- Agile yapıda uzmanlığa ve üretime değer vermeniz gerekiyor. Emeğe hakkını vermek lazım. "Birileri çalışsın, ben de onu alırım anlatırım" gibi bir yapı yok. İşi yapan kişi anlatır, onu kendisi yönettiğini bilir ve gitmeye çalıştığı amaç uğruna yaşar. Dolayısıyla uzmanlığın önemi Agile çalışma şeklinde çok fazla. Uzmanlık ve yetkinlikler ile yükselen bir yer. Bunun riski de şu; sürekli uzman ve yetişmiş iş gücüne ihtiyacınız var. Uzmanlığı kendi içinde sağlamanız lazım. Dolayısıyla tüm kariyer yolculuğunu da yetkinliklerin gelişimini teşvik edecek şekilde oluşturmanız gerekiyor.

- Peki, başlangıcı daha iyi yapabilir miydik? Şimdi bulunduğumuz noktadan geçmişe doğru baktığımızda; "daha geniş bir katılımla başlayabilseydik daha iyi olurdu" diyebiliriz. Tribe kuruluşunu, bahsettiğimiz üzere, kapalı ve küçük bir grup olarak yaptık. Ancak, bu işin özünde açıklık ve iş birlikteliği olduğu gibi katılımcılık da var. Dönüşümün daha açık şekilde, geniş çapta insanların Tribe'ın yapısı hakkında görüşünü alarak başlaması ve ilerlemesi içselleştirmeyi de kolaylaştırırdı. Deneyimleyerek görüyoruz ki herkes neyin neden olduğunu ya da olamadığını çok net anlayabiliyor. O yüzden, korkmaya hiç gerek yok. En baştan herkese açık bir tasarım süreci ilerletilebilir.

- Geriye dönük iyileştirme adına söyleyebileceğimiz diğer bir unsur da geçişi çok hızlı ve kısa zamanda yapmış olmamız. Ekipler genel olarak süreci destekleseler de yeni oluşacak organizasyon ve sorumluluklarla ilgili belirsizlikler, zaman zaman şüphelerin oluşmasına neden oldu. Ekibin nispeten genç olması, daha önce IT tarafında Agile yapıya alışık olmaları, Agile çalışma yönteminin temel prensiplerine hâkim olmaları işimizi kolaylaştırdı. Ancak, uzun süreli ve önemli bir geçmiş Agile deneyimimiz olmasaydı, bu hızlı dönüşüm ciddi problemler yaratabilirdi.

- Şu anda da kısmen halen devam eden, en fazla tartışılan konular; Product Owner ve Chapter Lead rollerinin sorumluluk alanları, performans yönetimi, bundan sonraki kariyer patikaları oldu. Bunları başta tam olarak netleştirmemiş olmak bizleri zorladı diyebiliriz. Ancak, biz de ekip arkadaşlarımızı dinleyerek onlardan gelen ihtiyaçlara yönelik olarak yapımızı Agile bir şekilde evrimleştirerek ilerliyoruz.

- Yöneticilerin çevik liderlere dönüşümünü önceliklendirin. Yöneticilerin bu yeni çalışma şeklinde doğru birer rol model olması önemli. Takımların güçlenmesi ve hızlı hareket edebilmesi için liderlerin destekçi ve yol açıcı olması gerekiyor.

- Bir diğer önemli konu da Agile prensiplerin tüm ekosistemde anlaşılması ve benimsenmesi. Bu yaklaşımı Tribe'ın ve Agile yapı içinde çalışanların benimsemesi o kadar kolay ki. Çünkü Agile'daki prensipler insanın doğasına uygun. Adı üstünde, birlikte çalışma. Tribe zaten insan doğasından, kökeninden gelen bir şey. Birlikte çalışalım, birlikte üretelim. Ortak hedef için durumsal liderlik var. Hasır örmede iyi olan, hasır örme işinin lideridir; avcılıkta iyi olan, avcılığın lideridir ve bu, insanların işine yaradığı için o kişi liderdir. Yoksa, yetkilendirildiği için değil. Bu durum birçok insanın içgüdüsüne ve doğasına uygun. Dolayısıyla Agile yapının içinde yaşayanlar bunu çok hızlı benimsiyor. Ama asıl konu, bunu tüm ekosisteme anlatmak ve herkesin bunu benimsemesini, bu yeni bakışa adapte olmasını sağlamak. Bu, çok önemli. Diğer türlü birbiriyle zıt mesajlar ortaya çıkmaya başlıyor ve o da sağlıklı değil. Bu değerlerin tüm ekosistemde, tüm şirkette anlatılması ve benimsenmesi lazım[3].

- Tüm şirketin Agile yapıyı anlaması ve mümkün olduğu kadar yüksek seviyede pratik edebilmesi önemli. Ancak, büyük çaplı değişimleri küçük parçalar halinde

[3]Agile Collage
Agile yapının içinde olan ya da olmayan tüm Vodafone çalışanları için temel Agile eğitimleri düzenledik. Buradaki amacımız sadece Agile'ın ne olduğunu anlatmak değil; diğer yandan yapılar Agile'a dönüşmese bile faydalı olabilecek noktalarda Agile pratiklerin organizasyona yayılmasını sağlamaktı. Böylelikle daha büyük bir dönüşümün kültür ayağı için tohumlarımızı ektik. Agile yapıdan yolu geçen geçmeyen herkes Agile Collage'a en az bir kere uğradı.

gerçekleştirmek üzere plan yapın ve planların kesinlikle değişeceğinin bilincinde olarak dönüşümünüzde ilerleyin. Acele etmeden, deneyip iyileştirerek dönüşüm yolculuğunuzu ilerletin.

- Eğitim önemli. Önce Agile felsefesini tanımak, tanıtmak gerekiyor. Bunun için eğitim tek başına yeterli değil, koçluk alınması da şart. Biz iç Agile Koç'larımızı yetiştirme konusunda geç kaldık. Bu yatırımı en baştan yapmanızda büyük yarar var.

Gelecekte Bizi Bekleyenler

Vodafone Türkiye olarak topyekûn dönüşüme yelken açtık. Altı yıl önce IT'de başladık. Bu başarı büyüdü ve bugüne baktığımızda bireysel iş birimi dönüşümü bitti, müşteri hizmetleri başladı, kurumsal iş birimi çalışılıyor. Epey yol kat etmiş olmamıza rağmen biz hala bu yolculuğun başlarında olduğumuzu düşünüyoruz. Mevcut işimizi ilerletiyoruz, daha hızlanıyoruz, daha sık müşterilerimiz ile konuşuyoruz, daha kaliteli çıktılar veriyoruz, daha hızlı yanılıp daha çabuk iyileştirebiliyoruz. Böylece de iş sonuçlarımız gelişiyor.

Önümüzdeki dönemde Vodafone'da Agile çalışma şekli nereye evrilecek diye bakarsak, orta vadede üç ana konumuz var;

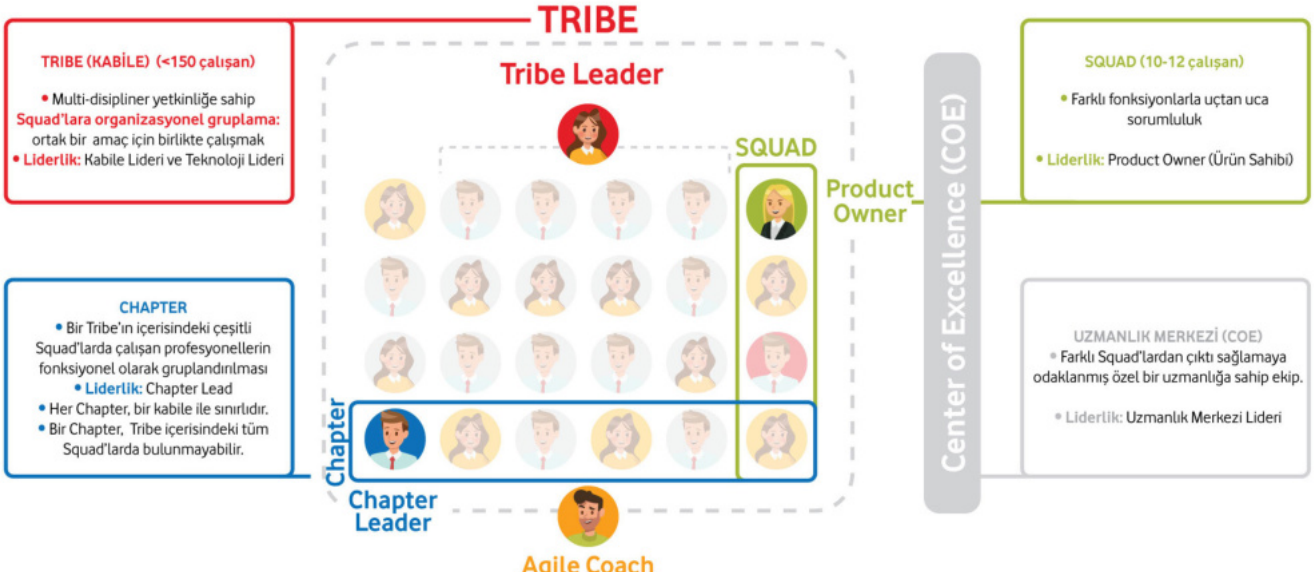
- Birincisi Chapter yapımız. Şu an uzmanlık kadrolarımız yetişiyor. Chapter Lead'lerimizin yükünün belli bir kısmını Chapter içinde yönetebilecek hale geleceğiz. Mesela, uzmanlık alanını oluşturmak, standartların gelişimini sağlamak, bilgi birikimini güçlendirmek gibi konular Chapter içinde üyelerin kendi kendine organize olmasıyla birlikte daha kolay ilerletilecek. Chapter'daki yetkinlik güçlü kişiler tarafından desteklenerek, gerçek bir topluluk bilinciyle bunlar geliştirilecek.

- İkinci konumuz ise bağımlılıkları adım adım çözmek olacak. Takımlarımızı odaklanma ve izolasyon noktasında daha etkin bir kurguya kavuşturacağız. Bunun en kritik tamamlayıcılarından biri de Core IT tarafında bu dönüşümün tüm katmanlarda devam ettirilmesi. Birlikte çalıştığımız tüm üçüncü parti paydaşlarımızı da hesaba katarak, bu dönüşümü IT'nin her kademesinde hızlandırarak devam ettirmemiz ve esnek yapılara sahip olmamız çok önemli.

- Üçüncü konumuz kültür. Kültür, başlı başına bir konu; ancak, eşgüdümlü çalışabilme ve takım olmaya yönelik kaslarımız üzerinde ilerlemeye devam edeceğiz[4]. Deneyerek iyileştirme ve sürekli öğrenmeyi de kültürümüzün ana unsurları haline getirerek güçleneceğiz.

Daha büyük resimden bakacak olursak, bundan sonraki aşamada, bu dönüşmekte olduğumuz yeni iş yapış tarzı yaklaşımının ticari modelimizi de değiştirmesini bekliyoruz. Örneğin kurduğumuz Squad'lardan bir tanesinden öyle güzel bir fikir gelecek ki, belki o Squad bir şirkete dönüşecek ve şirkete dönüşünce de zaten o şirket Agile doğmuş olacak. O zaman Agile dönüşümden bahsetmeyeceğiz. Agile bir yapı kendini bir şirkete çevirmiş olacak. Böylelikle Agile yaklaşımın iş modelimizi dönüştürüp yepyeni iş modelleri ortaya çıkardığına tanıklık edeceğiz.

[4]Let's Agile!
Takımın gerçek bir takım gibi hissetmesi çok kritik. Bunun da yolunun takım üyelerinin birbirlerini çok iyi tanıması ve gönülden bir bağ kurmasından geçtiğini düşünüyoruz. Bunun için özel bir Takım Koçluğu programı hazırladık; ismi de "Let's Agile!". Tüm içeriği Agile çalışma şekli baz alınarak dizayn edilmiş bir günlük takım çalıştayı. Ekibe geri bildirim verme, etkin iletişim, takım içi bağların güçlendirilmesi gibi bazı konularda vakalar üzerinden pratik araçlar kazandırıyoruz.


TRIBE
TRIBE (KABİLE) (<150 çalışan)
• Multi-disipliner yetkinliğe sahip
Squad'lara organizasyonel gruplama:
ortak bir amaç için birlikte çalışmak
• Liderlik: Kabile Lideri ve Teknoloji Lideri
Tribe Leader
SQUAD
Product Owner
Center of Excellence (COE)
SQUAD (10-12 çalışan)
• Farklı fonksiyonlarla uçtan uca sorumluluk
• Liderlik: Product Owner (Ürün Sahibi)
CHAPTER
• Bir Tribe'ın içerisindeki çeşitli Squad'larda çalışan profesyonellerin fonksiyonel olarak gruplandırılması
• Liderlik: Chapter Lead
• Her Chapter, bir kabile ile sınırlıdır.
• Bir Chapter, Tribe içerisindeki tüm Squad'larda bulunmayabilir.
Chapter
Chapter Leader
Agile Coach
UZMANLIK MERKEZİ (COE)
• Farklı Squad'lardan çıktı sağlamaya odaklanmış özel bir uzmanlığa sahip ekip.
• Liderlik: Uzmanlık Merkezi Lideri

Squad Nedir?

Squad, ortak bir amaç için bir araya gelmiş, farklı uzmanlıkları, ancak aynı hedefi olan çalışanlar topluluğudur. Agile çalışan bir şirket ya da Tribe, Squad'lardan oluşur. Her bir Squad'ın ayrı ve kendine has bir hedefi vardır ve bu hedefe ulaşmak için otonom bir şekilde çalışır. Hedeflere ulaşmak ve iş sürekliliğini sağlamak için elzem tüm alan ve rollerden en az birer üye olması beklenir. Üyeler zamanla birbirlerine yardımcı olup gereken durumlarda işlerini devralabildikçe, gelişen yetkinliklerle birlikte takım güçlenerek ilerler.

Kısaca, Agile pratikleri uygulayan, multi-disipliner yetkinlikteki, bir odak etrafında çalışan takımlara Squad diyoruz.

Squad'ın İlk Günleri

Farklı takımlardan Geliştirme Takımı üyelerine "Squad kurulduğunda ilk neler hissetmiştin" diye sorduk:

Her organizasyonel değişiklik, etkilenen çalışanlar için beraberinde bir belirsizlik ve heyecan da getirir. Agile çalışma yapısına ilk geçtiğimizde bu sadece bir ekip değişikliği değil, aynı zamanda hayatımın en az üçte biri olan çalışma hayatıma yepyeni bir bakış anlamına da geliyordu. Bu da benim için yeni şeyler öğrenecek olmanın getirdiği merak ve mutluluk hislerinin yanında çok daha büyük bir belirsizlik ve endişeye sebep oldu. Agile yapısına geçmeden katıldığım iki ayrı eğitimde Agile çalışma şeklinin pazarlama alanında henüz yeni yeni uygulanıp, pek örneği bulunmadığı için IT ve yazılım ekiplerinden örnekler verilmişti. Bu da bende Agile yapının pazarlama alanında nasıl çalışabileceğiyle ilgili şüpheler oluşturdu. Pratiğe geçene kadar da şüphelerim devam etti. Scrum Tahtası'nın karşısına geçip ilk günlük toplantımıza başlarken "Şimdi ne yapacağız?", "Şimdi ne olacak?" diye düşünmekten işlere odaklanmakta zorlanmıştım. Toplantı sonunda masamda üzerime aldığım işler ve kafamda soru işaretleri dolu bir şekilde oturuyordum. Açıkçası ilk başlarda mevcut çalışma şeklinin sadece adının değişip eski şekilde devam edeceğimizden de endişeliydim. Şu anda Squad'ımızın kurulduğu ilk günden beri yaklaşık 7 ay geçmiş durumda. Uzun yıllardır, hatta çalışma hayatımın başından beri Agile yapıdaymışım gibi hissediyorum. Bu çalışma şeklinin pazarlama alanına uygulanamayacağı şeklindeki ilk düşüncelerimin ne kadar yanlış olduğunu fark ettim. Özellikle bu şekilde bambaşka olan çalışmanın faydalarını gördükten sonra, şimdi Agile yapıda olmayan başka bir ekibe veya şirkete geçmem gerekirse 7 ay önce taşıdığım endişeleri tekrar yaşayacağımı düşünüyorum.

Eğitim aldığım her kurumda duyduğum Agile iş yapış şekline geçeceğimiz için bir hayli heyecanlıydım. Fakat yaptığım işi Agile bir yapıya nasıl getireceğimi tam olarak kafamda canlandıramamıştım. İlk Squad'ımda bazı rutin işleri bölmek ve tamamlamak daha kolaydı. Fakat yeni ekibimde her Sprint'te yeni şeyler üretmemiz ve geliştirmemiz gerekiyordu. Ekip ile yeni fikirler ürettikten sonra geliştirme aşamasını küçük görevlere bölmenin zor olmayacağını düşünmüştüm Şu andaki hislerim ilk günkünden çok da farklı değil. Bazı noktalarda ekipçe zorlanıyoruz. Ancak, Vodafone'da 2 farklı Squad'da çalıştıktan sonra bir iş nasıl Agile bir yapıyla ele alınır, bunu anlayabiliyor ve bu doğrultuda aksiyon alabiliyor olmak çok güzel bir duygu. Her gün daha iyiye doğru gideceğiz diye düşünüyorum.

Squad'ın kurulduğu ilk gün mevcut ekibimdeki arkadaşlarımdan ayrılıp daha önce pek tanımadığım ve farklı birimlerden gelen insanlarla bir araya gelmiştim. Eskiden aynı ya da benzer işleri yapan insanların bir araya gelerek oluşturduğu bir ekipte yer alıyordum. Dolayısıyla; birbirimizin derdini daha iyi anlardık, birbirimize daha çok yardımcı olabiliyorduk. Aynı bakış açısından bakıp konuşabiliyorduk.

Yeni yapıyla birlikte her birimiz farklı ekiplerden bir araya gelerek bir ekip oluşturduk. Bakış açılarımız farklıydı. Ben teknik konularla ilgili bir şeyler söylediğimde çok fazla anlaşılmıyor, onlar ticari olarak bir şeyler söylediğinde ben pek anlamıyordum. Ben ara katman, Web Servis, Veri Tabanı (DB) vb. diyordum; onlar ciro, DB (yıl sonu hedefi) vb. diyorlardı. Bir ara söyledikleri DB ile benim söylediğim veri tabanının aynı olduğunu bile düşünmüştüm. Ama zaman içinde hepimiz birbirimizin dilinden daha fazla anlamaya başladık. Yeri geldiğinde ben yazılım geliştirici olarak yapılan işin cirosu hakkında yorum yapabilirken yeri geldiğinde diğer arkadaşlarım da ara katmandaki "retry" süreçlerinden, web servis parametrelerinden bahseder olmuştu. Böylelikle aslında bir işin uçtan uca hangi süreçlerden geçerek hayatımıza girdiğini, ya da başka bir ifadeyle müşterinin hayatına girdiğini öğrenme imkânı kazandık.

Squad'ın kurulduğu ilk gün hem heyecanlıydım hem de biraz kaygı hissediyordum. Kaygı hissetmemin sebebi Squad'ımdaki insanları tanımıyordum, yapacağım işe de çok aşina değildim ve daha önce Agile pratiklerle ilgili bir çalışma deneyimim olmamıştı. Zamanla yaptığım işe hâkim olmaya başladım ve ekip arkadaşlarımı sevdim. Squad'ımda kendimi güvende hissediyorum.
Vodafone'da Digital Product Lead olarak çalışıyorum ve Vodafone Yanımda'nın faturasız hatlar tarafı ile ilgileniyorum. Bu iş birçok farklı yetkinlikten insanın bir arada çalışmasını gerektiren bir iş.
Dışarıdan bakıldığında bu kadar kişinin bir arada çalışması çok zor gibi durabilir. Ancak, Agile'ın bir şekilde bizi birleştirdiğini ve çalışmamıza olanak sağladığını düşünüyorum.

Alışma Süreci

Bu yeni çalışma şekline alışma süreci senin için nasıldı?

En başta hep beraber bir şeyler deneyimleyeceğimizi ve alışmamın kolay olacağını düşünüyordum. Sonrasında bu değişime herkesin aynı oranda katkıda bulunacağını düşünmemin biraz iyimser bir bakış açısı olduğunu fark ettim. Ritüellerin ve bu yeni çalışma şeklinin bizi sadece yavaşlatacağını düşünen ve diğerlerini negatif anlamda etkileyen üyeler oluyor. En büyük zorluk, takımdaki kişileri bu değişimin uzun vadede bizi hızlandıracağına inandırmak. Bu anlarda Chapter Lead'imizin ve Product Owner'ımızın "sürece güven" temalı davranışlarının büyük etkisi oluyor.

Agile çalışmaya alışma süreci, deyim yerindeyse, "Agile" bir şekilde ilerliyor. Agile yöntem uzun vadeli planlar yerine daha küçük adımları deneyerek daha hızlı geliştirmeyi öğütler. Takım olarak birçok sorunla karşılaşıyoruz, farklı denemelerde bulunuyoruz. Burada bir bireyden çok, öğrenen bir organizasyonun parçasına dönüşüyorsunuz. Böylelikle zorluklar karşısında bir takım olmak çözümü kolaylaştırıyor.

Yeni modelde çalıştığımız insanlar ve çalışma şeklimiz birlikte değişti. Başta tabii ki biraz zorlanıyor insan. Sanki iş değiştirmişsiniz gibi bir hissiyat oluşuyor hatta. Ama; gerek birbirimizi daha iyi tanıyıp kaynaşmak ve ekip olabilmek adına

yaptığımız çeşitli etkinlikler, gerekse Agile ve Scrum üzerine verilen eğitimler ve koçluk hizmetleriyle çok hızlı bir şekilde hem birbirimize adapte olduk hem de değişen her şeye.

Agile'ın yanı sıra dijital de bana göre çok farklı bir dünya ve öğrenmem gereken birçok terim var. Squad'a ilk başladığımda daha çok zorlanıyordum, ancak şu an kendimi her geçen gün işe daha hâkim hissediyorum. Benim en çok zorlandığım konu Ürün İş Listesi yönetimi oldu. İşleri herkesin anlayabileceği netlikte detaylı bir şekilde hep birlikte ortaya koyabilmeyi zamanla alışkanlık haline getirebildik.

Agile'ın En Pozitifleri?

- Yeni yapıyla birlikte hata yapmanın biraz daha kabul edilebilir olduğunu hissediyorum. Bu bence Agile çalışmanın çok olumlu bir yönü. Bir Sprint boyunca uğraştığımız bir işin olmadığını anladıktan sonra farklı bir şekilde yapmayı deniyoruz. Deneyip öğrenerek ilerliyoruz.

- Agile yapıda hızlı aksiyon alabilmek ve işleri küçülterek sürekli teslimat yapmak ekibin motivasyonunu yükseltiyor[5].

- Agile çalışmak hem esneklik hem de şeffaflık açısından daha iyi. Squad'larda birçok farklı uzmanlığa sahip kişiler bir arada çalışıyor. Herhangi bir yerde bir sorunuz olduğunda ya da tıkandığınızda kontak arama derdiniz yok, geç döndü derdiniz yok. Hemen yanı başınızdaki arkadaşınız olduğu için hızlıca istediğiniz bilgiye/yardıma ulaşabiliyorsunuz. Böylelikle bir işin bitmesi için geçen süre de oldukça kısalıyor ve optimum bir hale geliyor.

- Farklı profil ve bakış açısına sahip insanlarla birlikte çalışmak çok besleyici ve öğretici.

- Bir işin başlangıcından bitişine kadar söz sahibi olmak, işin şirkete ve aboneye olan etkisini sadece kendi uzmanlık alanından değil, diğer alanlardan da görebilmek motive edici.

- Squad içinde çalışmanın bana göre en güzel yanı kişiyi T-Shaped[6] yetkinlik setine uygun hale getirmesi. Dijital tarafta uzmanlaşmak istiyorum, ama aynı zamanda ticari bakış açımı da kaybetmemek istiyorum. Aynı zamanda daha önce hiç bilgimin olmadığı yazılım tarafını da öğrenmek istiyorum. Squad'ımda tüm bu işler yapıldığı için ve ben de hepsine bir şekilde katkıda bulunduğum için beni bu alanda geliştirdiğine inanıyorum.

- Tüm ekibin bir arada, aynı masada oturması iş aktarımlarının daha hızlı, titiz ve daha doğru olmasını sağlıyor. Bununla birlikte takımın tüm üyelerinin aynı hedefe sahip olması, bir tarafın talep yapan, diğer tarafın talep karşılayan partiler olduğu çalışma şeklinden çıkıp, bunu herkesin aynı hedef doğrultusunda, sadece farklı alanlarda efor sarf ettiği bir yönteme dönüştürdü. Bu sayede ekiplerin tüm üyeleri daha iyi sonuçlar için kendinden bir şeyler katmaya çalışıyor.

- Takım olarak çalışmanın iş sonuçları dışında moral ve motivasyon olarak da katkısı oldu.

[5]Bu hız, ekip sayısıyla orantılı. Ekip büyüdükçe, özellikle de kişi sayısı 10'un üzerine çıktıkça aksiyon almak yavaşlıyor.

[6]T-Shaped yetkinlik; kişinin bir konuda derin uzmanlığa sahipken ilgili çevre konularda da geniş bir perspektifte beceri ve anlayış içerisinde olduğunu ifade eder.

Herhangi bir problemle karşılaştığımızda hem takım üyelerinin hem de Chapter Lead'lerimizin bize yardımcı olmak için her zaman yanımızda olacaklarını bilmek bize güç ve moral veriyor. Bir sorun veya ihtiyaç durumunda kimse bir kişiyi suçlamıyor ve takım bu ihtiyaca birlikte karşılık bulmaya çalışıyor.

- Yaptığımız her bir işin hangi amaçlara hizmet edeceğini bilmek, bu işleri yaparken bizim daha da motive olmamızı sağlıyor. "Bunu neden yapıyorum ki?" sorusunun takımımızdaki hiçbir üye tarafından sorulmadığına eminim.

- Güne başlarken günün nasıl gideceğini konuşmanın da üretkenliği artırdığına inanıyorum.

Agile'ın En Zor Tarafları?

- Ritüellerin takım tarafından benimsenmesi. Özellikle fazla takım üyesi olan Squad'larda ritüellere katılım ve katkı düşük oluyor. Ritüeller sırasında "Nasılsa herkes bir şey söyler geçer biter. İşimi yapmaya dönerim" diye bekliyor insanlar. Fakat bu yüzden, bazı üyeler birbirini dinlemediği ve paylaşımda bulunmadığı için, şeffaflık anlamında sorunlar oluşabiliyor.

- Genelde Squad'larda her profilden bir kişi çalışıyor ve o kişinin izin alması gerektiğinde o konuda durmak zorunda kalabiliyorsunuz. Ama zamanla ekip için bilgi aktarımlarıyla bu tip durumlar da aşılıyor.

- Squad'ların kalabalık olması zor yanlardan birisi. Takım olarak bir araya gelmekte, bir çalışma yapmak için saat bulmakta bile bazen sıkıntı yaşayabiliyoruz. Sayıca kalabalık olduğumuz için ufak konularda fikir ayrılığı yaşayabiliyoruz.

- Yeni yapıda çok fazla toplantı olduğunu düşünüyorum. Bazen toplantı yapmaktan çalışamıyor hale geliyoruz. Agile rutinlerimizin üstüne diğer ekiplerle yapılan toplantılar eklenince bazen çalışmak için vakit bulmak çok zorlaşıyor.

- Tribe olarak Agile şekilde çalışsak da işlerimizi devam ettirebilmek için Agile olmayan farklı ekiplerle de çalışıyoruz. Farklı çalışma modeline sahip olmamız birbirimizi anlamamızı zorlaştırabiliyor.

- Bazı Squad'ların kalabalık olması takım olmayı güçleştiriyor. Bu takımların belki de bölünmeleri gerekiyor.

- Küçük taleplerin karşılanması eskisine göre zorlaşmış durumda. Tribe'ımızda birlikte çalıştığımız ekiplerden daha önce çok az vakitlerini alacak bir talepte bulunduğumuzda en fazla yarım gün içinde cevap alabiliyorduk. Şimdi, bu işin bir sonraki Sprint'te planlanmasını beklemek durumunda kalabiliyoruz.

- Önceki çalışma biçimimizde yaptığımız işi her anlamda geliştirme konusunda yeni fikirler bulmaya çalışıp, bunları en kısa sürede hayata geçirmeye çalışıyorduk. Anlık aklımıza gelen fikirleri hızlıca değerlendirip üzerine çalışmaya başlayabiliyorduk. Agile çalışma şeklinde önceden belirlenmiş bir önceliklendirilmiş iş listesinin bulunması sebebiyle yeni fikrin değerlendirilmesi için bir sonraki veya daha sonraki Sprint'lerin

beklenmesi gerekiyor.

- Retrospektif! Kimse o hafta kötü giden bir iş veya olay hakkında konuşmak istemiyor. Ritüellerin çoğunda "Çok iyi gidiyoruz işte, bitirelim gidelim" havası hâkim. Böyle anlarda Product Owner, Scrum Master ve Chapter Lead'in ilk atılımı yaparak ekiplere öncülük yapmaya çalıştıklarını gördüm. Bu rol model davranışlarla birlikte birkaç yorumdan sonra ekibin geri kalanının da katkı vermeye başladığını deneyimledik. Ancak, halen sıkıntıları açıkça konuşmakta zorlanıyoruz.

Squad'ınızda İyi Yaptığınız Pratikler?

Bizlerle paylaşabileceğiniz iyi pratikler var mıdır?

Müşteri odağı en yüksek ekiplerden olduğumuzu düşünüyorum. Chatbot ekibi olarak her gün kullanıcıyı dinleyebileceğimiz bir portale sahibiz. Ayrıca müşterinin sesi olan başka kanallardaki analizleri de dikkate alarak küçük ama değeri yüksek hızlı aksiyonları belirleyebiliyoruz. İş listemizi düzenlediğimiz Ürün İş Listesi İyileştirme aktivitemiz öncesinde müşteri davranışlarına ilişkin veriyi paylaşıp iş listemize aksiyon olarak ekliyoruz. Burada otonom çalışmaya yaklaştığımızı hissedebiliyorum. Yönetici seviyesindeki kararlara tamamen bağlı değiliz. Veriyi baz alarak kararlarımızı şekillendiriyoruz.

Bence Sprint Review sürecimizi iyi yapıyoruz. Paydaşlarımızı Sprint Review aktivitelerimize çağırıp yaptığımız işleri onlara mümkünse deneyimleterek gösteriyoruz. Böylece; bire bir abonenin deneyimini yaşamış oluyorlar. Örneğin; Vodafone Yanımda uygulaması içinde yer alan TL Yükleme sayfasına uygun olan aboneler için teklifler sunup sunulan bu tekliflere göre TL yüklemelerini sağlamak için bir entegrasyon yaptık. Bu entegrasyonu Review aktivitesine gelen paydaşlarımıza uygulama üzerinden gösterip bire bir kendilerinin deneyimlemelerini sağladık. Böylelikle; kullanıcı deneyimini aslında kendileri de yaşamış oldu ve geribildirimlerini daha doğru, sağlıklı bir şekilde verebildiler.

Retrospektiflerde bazen zorlansak da burada daima süreci iyileştirmeye yönelik kararlar aldığımızı düşünüyorum. Bazılarını hemen bir sonraki Sprint hayata geçiyoruz, bazılarını bir sonraki Retro'ya kadar çözememiş oluyoruz. Ancak yine de her Retrospektif sonrası kararlarımızı güncelliyoruz. Örneğin, Sprint Planlama'larda bir işi 15 dakikadan fazla tartışmamız gerekirse Ürün İş Listesi İyileştirme ritüelimizi iyi yapmadığımıza yönelik bir alarm olarak görüyoruz.

Product Owner'ımızın liderliğinde düzenli olarak strateji yol haritası toplantıları düzenliyoruz. Bu toplantılarla önümüzdeki iki çeyrekte hangi büyük işleri hangi sırayla tamamlamamız gerektiğini takımca kararlaştırıyoruz. Bu sayede takımın tüm üyeleri aynı sayfada oluyor ve aynı zamanda

aynı konularla ilgileniyoruz. Birbirimizi farklı konularda geliştirirken, farklı fikirlerle de projelerimizi zenginleştiriyoruz.

Squad olarak olmasa da Chapter olarak, Chapter Lead'imizle düzenli olarak "Chapter huddle" etkinlikleri düzenliyoruz. Bu sayede, farklı Squad'larda aynı role sahip çalışanlar olarak plan, proje ve gerçekleştirdiğimiz işleri birbirimize aktarıyoruz. Bu toplantılar sayesinde birbirimizi etkileyebilecek işlerden haberimiz oluyor. Bu işleri Squad'ımıza aktarırken de kullanabileceğimiz geliştirmeleri öğreniyoruz. Ek olarak, geniş bir gruptan farklı farklı yorum ve ö nerilerle de projelerimizi geliştiriyoruz.

Squad Ne Zaman Başarılıdır?

Başarı ölçütümüz zamanla takımların gelişmesi ve Agile pratiklerin öğrenilmesiyle beraber değişim gösterdi. İlk başlarda hem yeni bir terim olduğu hem de her Sprint'imizin sonunda ölçümlediğimiz için ilgili Sprint'te bitirdiğimiz iş oranını başarı olarak kabul ediyorduk. Sprint için planladığımız işleri zamanında bitirmek bir ölçüttü. İşi bitirmekten kasıt, uygun koşulların sağlanması diye özetlenebilir. Öyle ki iş listemizdeki öncelikli işler için kabul kriterlerimiz var. Bu kriterleri Product Owner'lar belirliyor. Böylelikle işleri planlanan Sprint'te, kabul kriterleriyle belirlenmiş şekilde bitirmek başarıydı diyebiliriz. O zamanlarda olduğu gibi şu anda da planladığımız her şey her zaman %100 bitmeyebiliyor. Fakat, o Sprint'in hedefine hizmet eden kritik işlerin tamamlanmasından bahsediyoruz. Sprint hedefini oluşturan ana işi tamamlayabildiysek o Sprint başarılı geçmiştir şeklinde bir bakış açımız vardı. Burada "Bitti Kavramından" da bahsetmekte fayda var. Yapacağımız işlerin genel kalite ve yapılış standardını takım olarak en baştan belirleyerek Sprint'ler koşmaya başladık. Bu doğrultuda Squad'ımızın başarısını değerlendirirken bu standartlara uyumumuz da önemli bir etkendi. En başta belirlediğimiz bu standartları zaman zaman ele alarak hâlâ iyileştirmeye çalışıyoruz.

Zaman ilerledikçe başarı kriterimiz ilgili Squad'ın iş hedeflerinin gerçekleşmesine dönüştü. Örneğin, orta vade hedefimiz dijital kanaldaki satış adetlerini yükseltmekse, ilgili bir Sprint'te satışların düşmesinin ana sebebini veya başka bir fırsat alanını bulup aksiyonlara başladığımızda o Sprint'i başarılı sayıyoruz.

Otonomi

Agile organizasyonda Squad'ların büyük ölçüde otonom, kendi kendine organize olabilen, yapılara dönüşmesi hedeflenir. Bu anlamda baktığında, otonomiyi nasıl değerlendiriyorsun? Karar alma gücü Squad'a ne ölçüde bırakılmış durumda?

Üst düzey kararlar Tribe seviyesinde alınsa da, müşteriye daha iyi dokunduğumuzu hissettiğimiz ve bunu verilerle kanıtlayabildiğimiz anlarda otonom kararlar aldığımızı düşünüyorum. Kullanıcıya çok yakın hissettiğimiz çağrı merkezi gibi kanallardaki sıcak konuları inceliyoruz. Analizin çıktılarını raporluyoruz ve bir aksiyon planı çıkarıyoruz. Tribe Lead "Neden bu konu üzerinde bu kadar zaman harcadınız ki?" diye sorarsa, analizimizin raporları yeterli oluyor. Yine de Tribe'ımızın genel hedeflerinden ve bakış açısından çok da dışarı çıkmamamız gerektiğini düşünüyorum. Stratejiyle hizalı bir şekilde ilerlememiz önemli.

Squad içinde bir konu ile ilgili hepimiz fikrimizi beyan ediyoruz. Birbirimizi ikna etmeye çalışıyoruz. Sonucunda bir karara vardıktan sonra hepimiz el birliği ile vermiş olduğumuz kararın gerektirdiği aksiyonları alıyoruz. Bunu yaparken de herhangi bir yöneticinin etkisi altında hareket etmiyoruz. Yer yer müdahaleler olsa da büyük ölçüde karar verme ve uygulama aşamaları ekiplerin kendilerine bırakılmış durumda. Böylelikle bir başarı olduğunda da, bir şeyler ters gittiğinde de yine bunu siz sahipleniyorsunuz. İnsan kendi karar verdiği işin başarısına da başarısızlığına da sitem etmiyor. Daha çok benimseyip, yanlış bir nokta varsa, fark ettiğinde, onu düzeltip tekrar deniyor. Böylelikle; işlerin verimi ve kalitesi artıyor.

Bu konunun Vodafone'da hâlâ gelişmeye açık yanlarının olduğunu düşünüyorum. Birçok işin karar alma gücü Squad'lara bırakılmış olsa da, bazı işler için bu durum geçerli olmuyor. Kritik görünen bazı işler genel çerçeveleri belli olduktan sonra yapılmak üzere Squad'lara geliyor.

Bu süreçte öğrendiğim önemli bir konu, bir Squad'ın otonom olabilmesi için iş yaptığı tüm partilerden en az bir üyeyi barındırmasıydı. Farklı Squad'ların veya Agile olmayan yapıların bir çalışanı, platformu veya sistemi ortak olarak kullanıldığında karşılıklı iş planları, öncelikler göz önünde bulundurulmak durumunda. Burada çıkacak fikir ayrılıklarında da orta yol bulucu olarak da Tribe Lead'lere bağlıyız. Squad'ımız yavaş yavaş çalışan sayısını artırdıkça, eksik rolleri kapattıkça, daha da bağımsız oldu. Şu an hedefimizi gerçekleştirdiğimiz sürece çoğu konuda otonom çalıştığımızı söyleyebiliriz. Ancak, özellikle stratejik ve portföyle ilgili konularda üst yönetim kararlarına bağlıyız. Örneğin, tarife fiyat ve içeriklerini güncellemek istediğimizde ilgili ekipten ve yöneticilerimizden onay almak durumundayız. Bu da aksiyonların canlıya ulaşma süresini uzatıp Agile yapımızı yavaşlatıyor.

Agile Çalışmanın Vodafone'daki Geleceğine Dair Kısa Kısa

Vodafone'da Agile'ın geleceğini nasıl görüyorsun?

Vodafone için orta vadede, eski çalışma sistemine sahip ekiplerin Agile bakış açısını kazandığı, gerçek anlamda içselleştirdiği bir çalışma ortamı görüyorum. Hangi ekipte olursam olayım Agile çalışma biçimini sürdürmek, ekip işleyişi özelinde bu çalışma şeklinden maksimum değer almak kişisel hedefim.

Çalıştığım şirkette böyle büyük bir dönüşüm başladığı için mutluyum. Ancak Agile'ın kurumsal bir şirketteki tüm departmanlara uygun olmadığını düşünüyorum, tabii bu benim kişisel görüşüm. Çalışanlar için eskiden şirket içinde uzmanlık alanı değiştirmek kariyerlerinde yükselmelerini yavaşlatan bir durumdu. Agile'ın bu olayı artık ortadan kaldırması gerektiğini düşünüyorum. Başka alanlarda uzmanlaşmak isteyen kişiler her açıdan desteklenmeli. Kişinin kariyerine olumsuz bir etkisi olmaması gerekiyor. Ayrıca şirket de her türlü eğitimle bu kişilere destek vermeli diye düşünüyorum.

Tüm birimlerin bu yapıya geçip Agile'ın daha da yaygınlaşması gerektiğini düşünüyorum. Yaygınlaşma sürecinde de farklı ekiplerin birlikte iş yapması gerektiği durumların organizasyonu konusunda özellikle dikkatli çalışılmalı ve bunun bir süreci olmalı. Bizde şu an bu kısım yavaş yavaş olgunlaşıyor. Farklı iki Tribe arasında organize olup iş yapmak istediğinizde dönem dönem sorunlar/gecikmeler yaşanabiliyor.

Vodafone Agile çalışma yapısına geçerek geleceği için önemli bir adım attı. Agile, Always On Marketing (analitik ve çoklu kanallı pazarlama) ve dijital dönüşümün çok hızlı olduğu bu dönemde işe yararlığını kanıtlamış durumda. Bu çalışma şekline olan güvenim, hem kendi ekibim hem de diğer Squad'lardan gözlemlediğim, zamanla artan iş sonuçları ve verimlilikle daha da sağlamlaştı. Bu doğrultuda henüz Agile yapıda olmayan ekiplerin dönüşüme dahil edilmesi, ekiplerdeki eksik rollerin doldurulması ve takımlara daha fazla güvenerek, onay mekanizmalarının azaltılmasıyla ekiplerin otonomluğunun daha da geliştirilmesine yönelik adımlar atılmalı.


Vodafone
Evde

Product Owner Kimdir?

- **Ekibin gideceği yönü belirleyen, Squad'ın vizyonunu ortaya koyan ve ekibi tek bir hedefe kilitleyen kişidir.**
- Müşterinin şirketteki temsilcisidir.
- **Product Owner, müşteriye değer katacak tüm ürün, servis ve deneyimleri müşteriye ulaştırma görevini üstlenen kişidir. Bununla birlikte müşterinin önceliklerini doğru anlayarak bu önceliklerle müşteriye sürekli değer yaratıldığından emin olur.**
- Product Owner, en küçük anlamlı parçayı (MVP) doğru belirleyip müşteriye sunduktan sonra, müşteri ile birlikte hareket ederek ve bunu doğru yönde geliştirerek üretilen değeri artırır.
- **Sahibi olduğu ürünün iş değerini ve ürünü kullanan kullanıcı topluluğunu temsil eden, ürüne ve bu ürünü kullanan kullanıcıya maksimum değeri yaratmak için Agile pratiklerle çalışan ürün geliştirme ekibinin hangi yol haritasını/öncelikleri takip edeceğini belirleyen takım üyesidir.**
- Ekiple beraber oluşturdukları değerli iş parçacıklarını müşteri etkisi, gelir beklentisi gibi getirileri göz önünde bulundurarak önceliklendiren ve bu anlamda Squad kapsamındaki iş ve hedeflere liderlik eden roldür.

Yeni Hayat ile Değişenler

Product Owner'larımıza sorduk, bir önceki görevin ve yeni görevin arasında ne gibi farklar var?

Önceki rollerime baktığımda, Product Owner rolümden sonra gördüğüm en büyük yeniliklerden biri farklı fonksiyonlarda çalışan arkadaşlarla beraber çalışmak ve onların iş planına liderlik etmek oldu. Hayatıma gelen diğer bir yenilik de işi tüm detaylarıyla planlamak ve uzun zaman içinde yapmaktansa parçalara ayırmak, etkisini görmek ve aksiyonları ona göre belirlemek oldu. Uzun süre ürün yönetimi yapmış biri olarak alıştığım yöntem her detayı en ince ayrıntısına kadar baştan düşünmek ve talep etmekti. Product Owner olduktan sonra fark ettim ki deneyerek ve sonuçlara bakarak detaylandırmak çok daha etkin ve uygulaması kolay bir yöntem.

Product Owner olduktan sonra hayatımdaki en büyük değişiklik, "Başlamayı bırak bitirmeye başla" felsefesi oldu. Müşteriye değer yaratan işler, öncelik sırasına dizilmiş ve küçük parçalara ayrılmış olduğunda ekibin odağını ve enerjisini doğru noktaya kanalize edebilmek en büyük yenilik ve odak noktam oldu. Her şeyi aynı anda yapmaya çalışıp ilerletmek yerine, odağı artırmak için gerektiğinde "Dur, önce bu, sonra diğeri" diyebilmek iş yapış şeklimi değiştirdi diyebilirim.

Product Owner'lık Vodafone ile hayatıma giren ve yaklaşık 7 aydır üstlendiğim bir rol. Önceki rollerimde ürün yönetimi ve ürün geliştirmeden sorumlu olduğum için aslında pazarı analiz edebilme, ürünün stratejisi ve hedeflerini belirleme, şirket hedefleri doğrultusunda alınacak aksiyonları belirleyip takip etme konularına hep vakıf oldum. Eski yapıda her ne kadar işin ana sahibi olsam da işin her aşamasına aktif olarak dahil olabildiğim bir sürecimiz bulunmuyordu. Ürün stratejisini belirleme, akışlarını netleştirme, tasarım ekibiyle ürünün tasarımını hazırlama süreçlerini önden gerçekleştiriyor, proje grubu kurulduktan sonra taleplerimin yapılabilirliğini ve zamanlamasını öğreniyordum. Konu yazılım aşamasına geldiğinde belki de aylarca yazılım ekibinden soru gelmediği takdirde aktif bir iletişim kuramıyorduk. Özellikle süresi uzun olan projelerde başta istediğimiz yapılana kadar pazardaki değişimler aylar öncesinde verdiğimiz talepleri karşılamamaya başlıyordu. İlk ilettiğimiz kapsam dahilindeki işler bitmeden yeni bir fırsatı proje ekibine getirmeye çalışmak genelde daha maliyetli ve ürünün pazara sunulma süresini daha da uzatıyordu. Agile organizasyon, bir bütüne ulaştıran parçaların da kendi içinde ne kadar değerli olduğunu anlamamı sağladı. İşi olabildiğince küçük parçalara bölmek, yarattığı değere göre en önemli parçasını hayata geçirerek ilerlemek hem pazardaki dinamizmi yakalamamızı sağlıyor hem de hata yapmışsak bile daha kısa sürede farkına varıp yeni aksiyonlar belirlememizi kolaylaştırıyor. Bu yapıyı eskiden olsa talepten vazgeçmek olarak algılardım. Halbuki en önemli noktaları en

hızlı şekilde hayata geçirmenin konforuna daha yeni ulaştım. Özellikle iş hayatında sonucu görerek tatmin olan bir insansanız, hangi rolde olursanız olun, süre uzadıkça projeye olan bağlılığınız azalıyor. Parçaları birleştirerek bütüne ulaşmak sizi bu süreçte çok daha motive ediyor.

Yeni rolümde ve iş yapış şeklimizde iş planlamamı çok daha net yapabiliyorum; ekip üyelerinin iş tanımlarını/yetkinliklerini daha açık görebiliyor ve onlardan beslenebiliyorum. Tüm bunlar daha motive çalışmamı sağlıyor.

Agile çalışma ortamında Product Owner olarak görev yapana kadar, iş hayatımda dijital kanallarda Ürün Müdürü olarak görev alıyordum. Agile dönüşümün bana en önemli etkisinin, projede yer alması gereken farklı yeteneklerin hepsinin bir arada çalışmasını sağlayan, takımın birbirini daha iyi anlamasını ve tek hedefe koşmasını sağlayan kültürün gelişmesi olduğunu söyleyebilirim. Bu kültür; projeyi, ürünü veya kanalı çok daha etkili yönetmenizi ve ciddi bir hız kazanmanızı sağlıyor. Öte yandan, geleneksel bir organizasyonda yöneticilik yapıyorsanız hem işin kendisini hem projeleri hem de size bağlı olan takımınızın yetkinliklerine ait dinamikleri yönetmekten de sorumlusunuzdur. Product Owner rolü ile gelen en önemli değişiklik-lerden birisi de bu sorumluluk alanlarının ürünün kendisine ve kullanıcıya kattığı değere odaklanmasını sağlamak. Artık, ürün geliştirme ekibinde yer alan farklı yeteneklerin yönetimi yerine, ürünün hedeflerine ulaşmasında takımın doğru yolda ilerliyor olup olmadığını daha yakından takip etmeye odaklanıyorum.

Bir önceki rolümle karşılaştırdığımda hayatımdaki en önemli değişiklik, üzerinde çalıştığımız ürünün uçtan uca takım olarak bizim sorumluluğumuzda olması ve başarıyı da başarısızlığı da tamamen şeffaf şekilde hissediyor olmamız. Daha önceden birçok projede çalıştım, uçtan uca projelerin her adımında oldum. Projeyi teslim ettiğimizde artık o iş bizden çıkıyordu. "Attığımız taş kuşu vurdu mu?" yoksa "Sadece proje mi teslim ettik?" gibi soruların yanıtını kurcaladıkça öğrenebiliyordum. Şu an ise sürekli yaşayan ve her an takip ettiğimiz ürünlerimiz var. Bir fidan gibi önce tohumun fide haline gelmesi, sonrasında palazlanması için her an gözümüz üzerinde. Hedeflerimizi değer yaratan küçük parçalara ayırmak ve buna rağmen resmin bütününü kaybetmemek, Agile organizasyonun benim için en önemli kazanımı.

Bir Sprint'in Hikayesi

Farklı Product Owner'larımızın gözünden bir Sprint'lerini inceleyelim:

Tipik bir çalışma haftamdaki en büyük odağım planlama oluyor. Squad için en doğru planlamayı yapabilmek için, öncelikle Ürün İş Listesi (Product Backlog) düzenlemesi-ne odaklanıyorum. Önden potansiyel bir işin müşteriye yaratacağı değeri anlayıp, bu konuda neyi nasıl yapabileceği-

mize karar verdikten sonra, onu nasıl parçalara bölebilirim diye düşünüyorum. Sonrasında parçalara böldüğüm işleri hayata geçirebilmek için nelere ihtiyacımız var, bağımlılıklarımız nelerdir, bunları anlamaya çalışıyorum. Takımla birlikte fikirleri olgunlaştırıp, Sprint'e alınabilecek hale getiriyoruz. Bir yandan da Squad'ımızın bağımlı olduğu partilerle görüşmelere başlayıp, onlarla yaptığımız plana uygun şekilde bu işleri öncelik sırasına diziyorum. Aynı zamanda vizyona gidiş yolunda nasıl bir Ürün İş Listesi hazırlamalıyım diye daha uzun vadeli planlar yapıyorum ve bu planı sürekli esnek bir şekilde yönetiyorum. Esnek Ürün İş Listesi yönetiminin Agile'ın hayatımıza getirdiği en anlamlı yeniliklerden biri olduğunu düşünüyorum. Müşteriye sunduğumuz ürünlere müşterilerin verdiği geribildirimleri düzenli ve bire bir temaslarla takip ediyorum, onlardan beslenerek Ürün İş Listemizdeki maddelerin içeriğini değiştiriyorum veya önceliklerini yeniden gözden geçiriyorum.

Sprint'in ilk haftasında çoğunlukla orta vade hedeflerimize ve buna yönelik planlara vakit ayırarak çalışıyorum. Önümüzdeki 3 ayın planını gözden geçiriyorum ve buna yönelik gerekli düzenlemeleri yapıyorum. Bu doğrultuda çalışmam gereken paydaşlarla bir araya geliyorum. Hedeflerimizin onların tarafından da anlaşılmasını ve benimsenmesini sağlamaya çalışıyorum. Bazen tam tersi şekilde paydaşlarım bana geliyor ve benden destek isteyebiliyor. Bu durumda görüşlerimi ve kendi planımızı onlara anlatıyorum ve ortak bir noktada uzlaşıyoruz. İşim Tribe içinde herhangi başka bir iş alanını etkiliyorsa, gerekirse Tribe Lead'imle ve Tribe'daki diğer ekiplerle bir araya geliyorum ve nasıl ilerleyeceğimize netlik kazandırıyoruz.

Sprint'in ikinci haftasında ise daha çok yakın dönemi detaylandırmaya odaklanıyorum. Bir sonraki Sprint için Ürün İş Listem üzerinde çalışıyorum. Hafta başında tüm ekip olarak Ürün İş Listesi İyileştirme (Product Backlog Refinement) toplantısında bir araya geliyoruz. Bu toplantıda aklımdaki fikirleri ekiple birlikte detaylı paylaşıyorum ve ekipçe farklı fikirleri değerlendiriyoruz. Buradan hep birlikte elimizde biraz daha net bir resimle çıkıyoruz. Ekibin görüşlerini ve önerilerini almak hem benim Ürün İş Listemi zenginleştirmemi hem de ekibin konulara hakimiyetini sağlıyor. Bu görüşme sonrası yapmam gereken bazı araştırmalar ya da görüşmeler ortaya çıkabiliyor. Bunları yapıp hafta ortası gibi ekiple yaklaşan Sprint'teki temel odak alanımızı kısaca paylaşıyorum ve fikirlerini almaya çalışıyorum. Ekibin yoğunluğuna göre burada aksamalar olabiliyor. Haftanın kalanında, Planlama Toplantısından önceki 2-3 günde kabul kriterlerimi detaylandırmış ve Ürün İş Listemin son halini netleştirmiş oluyorum. Tüm bu akışa paralel bir biçimde Sprint boyunca takım olarak yakın çalışıyoruz. İlgili Sprint'teki işlerin durumuna hemen hemen çoğu sabah benim de Daily Scrum'lara katılımımla birlikte beraber bakıyoruz. Gün içinde planladığımız aksiyonların beklediğimiz katkıyı verip vermediğini gözlemliyoruz ve gerektiğinde önceliklerde ufak tefek değişikliklerle hemen alternatif çalışmalar, deneyler yapıyoruz.

Çok eskiden beri Agile pratikler ile çalışan ekiplerden duyduğum bir kavram var: "sprint koşmak". O zamanlarda Sprint'in neden koşulduğunu merak ederdim, koşmaya başlayınca anladım. Bizim Sprint süremiz 2 hafta,

başka bir deyişle 10 çalışma günü. Bu sürenin 1-1,5 günü aslında Agile ritüelleri doğrultusunda yaptığımız toplantılarla geçiyor. Kalan gün sayımızda planladığımız işi bitirebilmek adına koşmamız gerekiyor. "İnsan sürekli koşamaz, yorulur" diyeceksiniz. Ama bu koşma, aslında bir yerlere yetişme derdi değil, adeta yarışta bitiş çizgisine yaklaşan bir atletin hissettiği gibi başarıya yakın olmanın yarattığı motivasyondan geliyor. Sprint içinde herkes koşarken, Product Owner olarak hem Sprint'in içinde hem de ilerisindeyim. Mevcutta üzerinde çalıştığımız işlerin beklentiyi karşılayıp karşılamadığıyla, hem de önümde koşmam gereken işleri önceliklendirme, netleştirme varsa bağımlılık noktalarını çözmekle ilgileniyorum. Aslında hem içeride hem de dışarıda olmak önümdeki işleri daha kolay planlayabilmemi sağlıyor.

Sprint dediğiniz döngü önce onu planlamakla başlıyor. Etkili bir planlama için ise, hedeflerinizi, hedeflerinize yönelik stratejinizi, bu stratejiyi destekleyecek yol haritanızı ve bu yol haritasına ait adımları nasıl önceliklendireceğinizi belirlemiş olmalısınız. Daha da ötesi, tüm bu hikâyenin takımınızın da içine sinmesi ve ürünün uzun vadede nereye gittiğini tüm takımın hissedebilmesi çok önemli diyebilirim. Bu anlamda "Neredeyiz?" ve "Nereye gidiyoruz?" sorularını değerlendirmek için takımla belirli aralıklarla bir araya geliyoruz. Peki bunun neye katkısı var? Öncelikler doğrultusunda tüm takımın hedefi özümseyerek bu hedefi destekleyecek kendi maddelerini oluşturmasını destekliyor. Böylece Sprint'in hedefini herkesin sahiplenmesini sağlıyoruz. Planlama sonrasında Sprint devam ederken Product Owner'ın gözü her zaman Sprint'in gidişatının takibinde oluyor. Bunun haricinde bir sonraki Sprint'in işlerini hazır etmek için tüm paydaşlarla bir araya geliyoruz ve Ürün İş Listesi İyileştirme ritüelimizi gerçekleştiriyoruz. Sprint içinde gerçekleştirilen her bir Ürün İş Listesi İyileştirme aktivitesi, gelecek Sprintlerin planlanmasına önemli katkı sağlıyor. Son olarak da Sprint ritüellerinin en önemli parçalarından biri olan Retrospektif'ten bahsedebilirim. Bu üzerinde yeterince durulmayan bir konu olabiliyor. Ancak, Sprint Retrospektif ritüelinde takımı, bakış açısını, neleri iyileştirebileceğimizi bizzat takımdan dinlemenin Product Owner'ın beklentisine ve iş yapışına önemli katkı sağladığını söyleyebilirim.

Product Owner'ın Yetkinlik Gelişimi

Product Owner'larımıza sorduk, kendini geliştirmek için neler yapıyorsun?

Agile yapı, bahsedildiği gibi yaşayarak öğrendiğiniz, doğru sonuca deneme yanılmalarla ulaştığınız bir organizasyon demek. Öğrenen organizasyon kültüründe hata yapılmasına izin verilir, ders aldığınız müddetçe. Farklı organizasyon kültürlerinden gelen kişilerin bu yeni kültüre alışması ve öğrenmesi biraz zaman alabiliyor. Kendi adıma söylüyorum, zorlandım. Ancak, şirket bu adımları kolaylaştırmak adına ekibin her bir üyesine temel Agile eğitimi veriyor.

Bu eğitim, Vodafone'a ilk başladığım zamanlarda tüm rol ve sorumlulukları anlamam adına son derece faydalı oldu. Bunun yanı sıra hem ekibin Agile pratiklere alışması, hem de ekip içi veya genel yaşadığımız sorunları çözmemizde bize destek olacak Agile Koç'larımız var. Bu sayede, aslında teorik olarak ulaşabileceğimiz bilgileri hayata nasıl geçireceğimiz ya da nasıl alternatif yollar üretebileceğimiz konusunda kendimizi yalnız hissetmiyoruz.

Agile'ın gücü, kendi kendine organize olan yetkin ekiplerden geliyor. Buradaki yetkin ekip içinse her bir bireyin farklı bakış açıları kazanarak T-Shaped beceri gelişimi önemli. Bu anlamda T-Shaped yapıyı destekleyen ekip içindeki eğitimleri kendimiz düzenliyoruz. Ben de ekip arkadaşlarımın deneyimlerinden beslenerek öğreniyorum. Örneğin, Karar Mimarımız (Decision Architect) iş yapış şeklini, kullandığı ekranları anlatan bir paylaşım düzenleyerek bizleri bilgilendirdiğinde, sadece Product Owner rolünü değil, Squad'ımız içinde diğer iş alanlarını da öğrenmiş ve kendimi geliştirmiş oluyorum.

Agile çalışılan bir organizasyondaysanız, bulunduğunuz rolde kendinizi geliştirmek için öncelikle Agile çalışma biçiminin dinamiklerini ve temel mantığını içinize sindirmeniz çok önemli. Ben bu role ilk başladığımda çok dinledim, okudum ve Agile Koç'larla her bir sürecin mantığını anlamak için çok zaman geçirdim. Bu anlamda, aklıma bir soru takıldığında, beni destekleyecek bir koçun her zaman orada olmasının çok büyük katkısı oldu diyebilirim. Öte yandan, Agile dediğiniz yapı, kesin kuralları olan bir süreç değil. Öyle olsaydı, varoluşuyla bir çelişki oluştururdu zaten. Siz Agile'ın temelinde yatan ana düşünceyi içinize sindirdiğiniz-de, takımınız ve ürün geliştirme süreciniz bunun üzerine şekilleniyor ve bir süre sonra Agile bakış açısında kendi doğrularınızı buluyorsunuz. Bir nevi, her Agile takım, temel dinamiklerin çerçevesinde kendine yakışanı giyiyor demek yanlış olmaz. Hem kendi rolüm için hem de takımın işleyişi için her zaman süreci daha nasıl iyileştirebiliriz diye denemekten çekinmedim. Agile'ın mottosu olan dene-yanıl bakış açısını rol ve sorumluluklarım için de gerçekleştirmeye çalışıyorum. Bunun hem kendi rolümün gelişmesine hem de takımın daha iyi bir noktaya gelmesine katkı sağladığını düşünüyorum.

Product Owner rolü yeni bir rol. Bu role adaptasyon ve kendimi en doğru şekilde hazırlamak için öncelikli yaptığım iş, Agile'ın genel felsefesini doğru özümsemek ve bu felsefenin bu roldeki karşılığı tam olarak nedir ve ben nasıl uygulamalıyım sorusuna cevap vermek oldu. Bunun için temel Agile eğitimleri alarak ve Agile ile ilgili yazılmış kitapları okuyarak başladım. Sonrasında kendi rolüm için daha detaylı eğitimler aldım. Şirketin sponsorluğunda düzenlenen Product Owner Yetkinlik Gelişimi eğitim serisi, rolümü içselleştirmemde ve uygulama noktalarını doğru anlamamda çok faydalı oldu. Ayrıca, Agile Koç'lardan uygulama sırasında sürekli geribildirimler almak ve gelişim alanlarımın farkına varmak da rolümde daha yetkin hale gelmemde çok katkı sağladı. Aynı zamanda doğru vizyonu oluşturabilmek için sektördeki ve diğer sektörlerdeki en iyi uygulamaları ve yeni trendleri sürekli takip ettim ve etmeye devam ediyorum.

Bu rolü hakkıyla yapmak için Geliştirme Takımındaki herkesin nasıl çalıştığını ve ne gibi adımlardan geçtiğini anlamak,

işin kendisine ve işi yapmak için gereken kabiliyetlere hâkim olmak gerekiyor. Bunun için her ekip üyesiyle yeterli zaman geçirmeye ve işlerini anlamaya çalışıyorum. Bu pozisyon kesinlikle müdürlük değil, Squad'daki kimse size raporlamıyor. Sırf bu yüzden, müdürlük etiketine değil, doğal liderlik becerilerine güvenerek ilerleyebilmek gerekiyor. Hem ekipten hem Tribe Lead'imden hem de diğer Product Owner arkadaşlarımdan geribildirim almayı, sorulmaya sorgulanmaya açık olmayı benimsiyorum. Liderlik alanında okuyorum, araştırıyorum. Ekibin dengesi olmak çok önemli. Başarıların paylaşılmasını, hataların da ekipçe sahiplenilmesini benimsemeye ve bu konuda takım arkadaşlarıma da örnek olmaya çalışıyorum. Kişileri tanımak için zaman ayırıyorum, beklentilerini ve zorlandıkları alanları biliyorum. Bu empati, daha sonra işleri planlarken ve ekipten destek isterken çok faydalı oluyor.

Etkileşimler ve Zorluklar

Agile organizasyonla birlikte hayatımıza yeni birçok rol girdi, senin rolün bunlardan sadece birisi. Peki, yeni yapıdaki diğer rollerle etkileşimlerin nasıl? Yeni yapıda yaşadığın zorluklar neler?

Product Owner, Tribe içinde kendi ürünü için vizyonu, Ürün İş Listesini, öncelikleri ve işlerin kabul kriterlerini belirleyen rol olarak konumlanıyor. Tribe içi ve Tribe'lar arası diğer Product Owner'lar, Chapter Lead'ler, Tribe Lead'ler ile sürekli etkileşim halinde olması gereken bir rol. Diğer Product Owner'larla etkileşim ve iş birliktelikleri daha çok birbirini etkileyen işlerin nasıl yapılacağını belirlemek sırasında oluyor. Bir Product Owner'ın önceliği bir diğerinin önceliğinde olmayabiliyor veya yapılacak olan işin içeriğine karar verirken farklı fikirler ve bakış açıları olabiliyor. En zor kısmı, bu bakış açılarını ortak noktaya getirip ortak bir karara varmakta oluyor. Böyle durumlar yaşandığında genellikle ilgili Product Owner'larla bir araya gelip, konuyu nasıl çözebileceğimizle ilgili önerilerimi paylaşıyorum ve onların önerilerini dinliyorum. Bunun sonucundaki ortak önerimizi uygulamaya alıyoruz ve iyileşmeyi takip ediyoruz. Ortak önerideki çalışmayan noktaları sürekli izleyip, Sprint'lerle öğrenerek geliştiriyoruz. Böyle zorluklar, anlattığım gibi, iletişimle bir yere kadar çözümlenebiliyor, öte yandan bu önerileri ortak paydaya getirmek, sürekli izlemek ve geliştirmek için ciddi zaman ve enerji harcanıyor, bu da işleri yavaşlatabiliyor. Bence bunların minimumda yaşanması için Squad içi otonomluğun mümkün olan en üst seviyede artırılması gerekiyor. Bunu sağlayabilmek için de sorumlulukların mümkün olduğunca birbirini etkilemeyecek şekilde izole edilerek Squad'lara dağıtılmış olmasının oldukça kritik olduğunu düşünüyorum.

Product Owner'ın iletişimde olduğu bir diğer rol de Chapter Lead rolü. Ben genellikle ihtiyaç duyulan rol ve yetkinliklerin Squad'daki sürekliliğini sağlamada Chapter Lead'lerin desteğini alıyorum. İşe alım, istenilen kalitede iş üretebilecek yetkinliklerin geliştirilme-

si, T-Shaped yapıya Squad'daki kişilerin adaptasyonunun sağlanmasında sürekli etkileşim halinde oluyoruz.

İki farklı rolü ele alırsak; Product Owner ekibin ne yapacağını, stratejiyi ve vizyonu ortaya koyuyor; Chapter Lead ise bu yolda takımlarda gereken yetkinliklerin geliştirilmesine yardımcı oluyor. Özellikle yeni organizasyona geçtiğimizde bu iki rol arasında gri alanlar oluşma ihtimali var diye düşünüyordum. Çünkü özellikle teknik olmayan tarafta yetkinlik geliştirmek için önceliklere ve aksiyonlara detaylı dahil olmak gerekiyor. Bu Product Owner'ın mevcut görevini sorgulamak veya yakından takip etmek anlamına gelebilir, rahatsız edici olabilir. Biz bunu çok yaşamadık. Tribe'da doğal olarak işlerimiz ayrıştı. Chapter Lead'ler genelde Squad'larda çözülemeyecek ve tüm Tribe'i ilgilendiren konulara ve yetkinlik geliştirmeye odaklandılar. Birbirini destekleyen ve birbirinin ayağına basmayan bir çalışma yöntemini benimsedik. Bence Agile ekiplerinde sorunsuz çalışmak kişilerin iyi niyetine ve Tribe Lead'in tutumuna da bağlı. Başka bir takımda olsaydık sorunlar yaşayabilirdik belki, diye düşünüyorum.

Product Owner rolündeki arkadaşlar olarak çoğunlukla iş birimlerinden geliyoruz, çoğumuz IT kökenli değiliz. Ekibimizde IT kökenli arkadaşlar var. Başlarda onları anlamak ve onlarla etkin çalışmak konusunda zorlandım; terminolojiden çalışma yöntemine ve çalışma hayatından beklentilerine kadar her alanda farklılıklarımız vardı. Bir sürü kritik yetkinliği başta ben bilmiyordum, öğrenene kadar muhtemelen aynı eksende konuşamadık. İş hayatından beklentilere gelince; örneğin iş biriminden gelen bir arkadaş için sunum yapmak, raporu üst yönetimle paylaşmak çok önemliyken, teknik geçmişten gelen bir arkadaş için bu bir külfet, hatta rahatsız edici bir iş olabiliyor. İşlerin sahiplenilmesinde hem istekleri anlamak hem de ekibin esnemesini sağlamak lazım. Ben teknik yetkinlikleri, ekip de benim beklentimi anladıkça, hep birlikte iyi işleyen bir iş yapma şeklinde buluştuk.

Product Owner'lar Tribe içinde stratejiyi belirleyen ve bu doğrultuda aksiyon ve hedefleri takip eden kişiler olarak konumlanıyor. Product Owner olmak hem ekibin içinde hem de ekibin dışa açılan yüzü olmak demek. Gerekli tüm paydaşlardan gelen talepleri değerlendirmek, işin faydasını analiz etmek, başka Squad'larla ortak çalışılması gerektiğinde her ekibi benzer paydada buluşturmak bir taraftan da ekibin üretkenliğini ve katılımını da besleyecek ortamı sağlamak konusunda Product Owner'lara ciddi sorumluluk düşüyor. Özellikle bir işi yapabilmek için başka bir Squad'a bağımlılığınız varsa, hem diğer Squad'ın önceliğinde olmayı sağlamak, hem de benzer tarihlerde çıktı verebilmek adına düzgün planlama yapmak, olmazsa olmazınız. Özellikle Product Owner'lığımın ilk dönemlerinde Ürün İş Listemdeki maddelerin Sprint'e almak için yeterince hazır olmaması korkusuyla karşı karşıyaydım. Bu gibi durumlarda ekibin de sesini dinlemek, belirlediğiniz hedef doğrultusunda ekipteki herkesin fikir ve projelere katılımını sağlamak, sizi ve ekibi daha başarılı sonuçlara götürüyor.

Product Owner, Tribe Lead'e raporlayan bir rol. Bu yüzden, Product Owner'ın önceliklendirdiği işlerde Tribe Lead'in desteğini alması çok önemli. Bunu

yapabilmek için Product Owner'ın şeffaf olması, işleri neye göre önceliklendirdiğini doğru şekilde anlatabilmesi, doğru bildiği konularda ısrarcı olması kritik diye düşünüyorum.

Diğer bir etkileşim içinde olduğumuz rol Chapter Lead. Chapter Lead ile olan ilişkilerde karşılıklı destekleyici bir yapıda ilerlemek önemli. Chapter Lead'lerin Squad'daki işleri engellemeyecek kadar uzakta, gerektiğinde ise destek için bir nefes kadar yakında olması değerli. İlk başta Chapter Lead'ler ile hangi konularda sorumluluklarımızın ayrışacağında küçük çaplı sıkıntılar yaşadık. Ancak, açık iletişimle birbirimizi anlayacak şekilde ilerledik ve şu an çok iyi bir sinerji yakaladık.

Önceliklendirdiğimiz işlerin ekip tarafından sahiplenilmesi konusunda Scrum Master'ların desteği son derece değerli. Onların firiklerini almak, açıkça beklentiyi ve hedefleri paylaşmak ve Ürün İş Listesinin sahipliğini ekibe indirgemek, hem sonuçlara ulaşmayı kolaylaştırıyor, hem de ekip içindeki motivasyonu artırıyor.

Agile dönüşüm yolculuğumuzda Tribe yapısının kurulmasında görev alma şansım oldu. Bu süreçte en önemli gözlemim Agile Koç'un ve Scrum Master'ların gerekliliği oldu. Squad'ın kurulumu öncesinde yapılan hazırlıklardan başlamak üzere ekibin eğitimi, ekibin kurulması ve yeni süreci uygulamaya başlaması, Sprint'ler sırasında yeni pratikleri desteklemesi, Product Owner'lar ve diğer rollerin hizalanması gibi her konuda Agile Koç'ların ve Scrum Master'ların katkıları çok değerli. Squad içinde takım ruhunu elde etme, işlerin sahipliği, şeffaflığın artırılması ve engellerin ortadan kaldırılması noktasında bu iki rol çok kritik. Bir gün Tribe kurmam istense ilk talep edeceğim çalışma arkadaşım Agile Koç olacaktır.

Öncelik Savaşları

Kimin önceliği en önce? Product Owner olarak Squad seviyesindeki öncelikleri belirlemek senin görevin. Peki, bu gerçekten olması gerektiği gibi çalışıyor mu?

Product Owner olarak sorumlusu olduğum ürünün yol haritasını ve önceliklerini belirleyebiliyorum. Bu durum, tamamen paydaşları kapının dışarısında bırakmak değil. Aksine, tüm şeffaflığıyla onlarla aynı noktada buluşmak. Her zaman her paydaşınızla aynı düşüncede, aynı öncelik seviyesinde olacaksınız diye ideal bir gerçeklik yok. Ayrı düştüğünüz yerler olacak ve olsun da. Bu zaten çeşitliliğin ve bakış açısının gelişmesini sağlayan en önemli etken. Önemli olan konu, neyi niye yaptığınızı öncelikle kendinizin ve takımınızın iyi anlamış olması. Sonrasında ise paydaşlarınızın ihtiyacını iyi anlamanız ve eğer bu ihtiyaca hemen karşılık

veremiyorsanız nedenini tüm şeffaflığıyla, bir savaşa dönüştürmeden aktarabilmeniz. Eğer bu ihtiyacın değer kattığını düşünüyorsanız, kendi önceliklerinizde değişikliğe gidebilecek açıklıkta olmanız da önemli. Bu bakış açısının Vodafone'da karşılığını bulduğunu gördüğüm için ben kendimi şanslı hissediyorum.

İşin iki boyutu var; ekip içi hizalanma ve ekipler arası hizalanma. İlki için bakacak olursak, eğer Squad'da çalışan arkadaşlar işi bildiğinize ve uzmanlığınıza emin olursa; siz de onlardan fikir almaya, gerekirse önceliklerde ve iş planında esneklikler yapmaya açık olursanız, yani önerileri "duyarsanız", sıkıntı olmaz. Ben Ürün İş Listesi İyileştirme toplantılarında ekibe opsiyonlar götürüyorum, farklı fikirlere açık olmaya çalışıyorum. Ekipten gelen önerileri de yaptığımız veya önceliklerde dikkate aldığımız zamanlar oldu. Onlar da bu sebeple belirlediğimiz öncelikleri benimsiyorlar. Bu yüzden hiç sorun yaşamadık diyebilirim.

Ekipler arasında hizalanmaya bakarsak, dışa bağımlı olma riskimiz olan durumlarda işi önden değerlendirip bağımlılıkları son dakikaya kalmadan iletişimle şeffaflaştırmaya çalışıyoruz. Yani ilgili diğer Squad'lara önceden haber vermeye ve onların önceliklerine zamanında girmeye çalışıyoruz. Ama sizin önceliğiniz diğer Squad'ın önceliği olmayabiliyor. Bu noktada duruma ve işe göre değişen yöntemler izliyoruz. Aslında baktığınızda, bu durum çoğunlukla Product Owner'lar arasında çözülüyor, nadiren ise Tribe Lead'lerin desteğini almamız gerekebiliyor.

Squad'ımla ilgili önceliklerim yalnızca bizim Squad'ı ilgilendiriyorsa, karışma veya müdahale çok nadir yaşanıyor. Ancak, birçok iş diğer Squad'ları da ilgilendirebiliyor. Bu durumda müdahale ile daha sıklıkla karşılaşılıyor diyebilirim. Bu durumlar genelde birden çok hizalanma toplantılarıyla sonunda çözümlenebiliyor; ancak bu durum hızlı ilerlenmek istenen işlerde yavaşlamaya sebebiyet verebiliyor. Bu yüzden, diğer Squad'lardan destek ihtiyacı olan işler ve getirdiği bağımlılıkların yönetiminin en büyük gelişim alanımız olduğunu düşünüyorum. Mevcutta bu durumu birden çok hizalanma toplantıları yaparak yönetmeye çalışıyoruz. Bu toplantılarda müşteri kavrayışı, buna istinaden verilerin bize ne söylediği gibi argümanlarla bu işin neden öncelikli olduğu veya olmadığı yönünde birbirimizi ikna ediyoruz. Bu sistem her ne kadar sonunda bizleri bir ortak noktada buluştursa da bu hizalanma sağlanana kadar geçen zaman ve enerji kaybı bizleri yavaşlatıyor. Bu konuda zamanla deneyim kazandıkça, bağımlılıkları minimuma indirgeyecek sistemler kuracağımızı ve böylelikle daha iyi bir noktaya geleceğimizi düşünüyorum.

Organizasyon içinde önceliklerin Product Owner'lar tarafından verildiği gerek Squad içinde gerekse paydaşlar tarafında biliniyor. Ancak, elbette ki herkes kendi talebinin öncelik alabilmesi adına daha çok söz sahibi olmak istiyor. Biz de başka Squad'ların önceliğine girebilmek için aynı durumu yaşıyoruz. Bu noktada ortak hedefler almak hem müdahale savaşlarını engelliyor, hem de ekipler arası daha sinerjiye dayalı iş birliklerinin yaratılmasına hizmet ediyor. Product Owner olarak bir talebi neden önceliklendirdiğimi ya da alamadığımızı, bir işi Sprint'e almadan önce işin hazır olma kriterlerini yerine getirecek seviyeye gelmesi gerektiğini olabildiğince açık bir şekilde ilgili tüm paydaşlarıma anlatıyorum. Bu noktada özellikle

Agile pratiklerle çalışmayan ekiplerin bizim çalışma şeklimize empati kurmalarını sağlayarak hayatımızı kolaylaştırmaya çalışıyorum.

Agile yapısında ekipler olabildiğince bağımsız iş yapabilir seviyede olmaya çalışır. Ancak gerek organizasyon gerekse teknik/bilgi birikimi açısından tamamen bağımsız olduğumuz bir yapımız maalesef ki yok. Özellikle teknik anlamda birçok Squad'ın dokunduğu sistemlere herhangi bir bağımlılığınız varsa o noktada öncelik almak son derece zor olabiliyor. Bağımlılıkları yönetebilmek için ekiplerin yetkinliklerini artırmak, bu anlamda bir bilgi birikimi kazandırmak son derece kritik diye düşünüyorum. Bu hususta özellikle işleri şekillendirdiğimiz Ürün İş Listesi İyileştirme toplantılarımıza her alanda destek sağlayabilecek ve farklı görüşleri bize kazandırabilecek çalışma arkadaşlarımızı davet etmeye çalışıyoruz. Ekipler kendi içlerinde otonom yapıda çalışabildiklerinde diğer ekiplere dokunan noktaları belli zaman aralıklarında yaptığımız tüm ekiplerin bir araya geldiği önceliklendirme toplantılarıyla daha kolay şekilde çözebiliyoruz.

Kesinlikle önceliklendirme Product Owner'ın işi, fakat bu önceliği verirken ekibin katkısı da önemli. Her zaman paydaşlardan kendi işlerinin daha öncelikli olduğu konusunda baskı alıyorum. Farklı takımlar kendi işlerine öncelik vermemiz için bazen sadece benimle kalmayıp Chapter Lead, Scrum Master ve ekibin diğer üyeleriyle de iletişime geçerek araya girmeye çalışabiliyorlar. Bu durumda ekiple tek ses olarak konuşup tüm bu taleplerin Product Owner'a yönlendirilmesini sağlamak önemli. Takım olarak bizim iyi olduğumuz bir konu, yani tek ses olarak hareket edebiliyoruz gerçekten. Bununla birlikte Tribe Lead'ın bu süreci sahiplenmesi, benim görüşlerime değer verip gerektiğinde aradan gelen müdahaleleri engellemesi işimi yapmamı kolaylaştırıyor.

Ancak müdahaleler dışında bağımlı olduğumuz ve bize bağımlı olan Squad'lar da bir gerçek, bunu da yönetmemiz gerekiyor. Bu anlamda Product Owner'ların aralarında hizalanması önemli bir konu. Birlikte Ürün İş Listesi düzenleme görüşmeleri yapıp bağımlılıklara istinaden beraber planlama yapmaya çalışıyoruz. Bu doğrultuda yaşanabilecek sıkıntıları büyük ölçüde en aza indirgiyoruz. Aslında bu noktada, yönetim anlamında bence Product Owner'lara önemli görev düşüyor. Product Owner kendi ürününe ve yol haritasına ne kadar hakimse ne kadar şeffaf bir zeminde iş birlikteliğine açıksa bu bağımlılıkların yönetimi o kadar kolaylaşıyor. Tabii, burada birlikte çalışma ihtiyacı olan, bağımlı olan Squad sayısı ne kadar artarsa işiniz o kadar zor. Mesela, altı Squad'ın görev aldığı bir projede yer alan Squad'lardan biriydik. Bu projede işlerin anlaşılmasından tasarlanmasına, geliştirilmesine ve test edilmesine kadar ciddi problem yaşamıştık. Tabii, bu bizim ilk geniş çaplı deneyimimizdi. Daha önce de birkaç Squad'ın bir arada çalışması gerektiği deneyimlerimiz olmuştu. Ancak, altı takımın bir araya gelmesi ve hiçbir takımın tek işinin bu birlikte ilerletilmesi gereken proje olmadığı bir senaryo çok farklıydı. Bu projeyi ciddi anlamda tıkandığımızı hissettiğimiz belirli bir noktada durdurduk. Agile Koç'u sürece dahil ederek tüm yapıyı tekrar ele aldık. Squad'ları, aralarındaki bağımlılıkları ve Squad dışından beklentileri tekrar ortaya koyarak çalışma yapımızı revize ettik. Böylelikle hangi Squad'ın hangi Squad'a bağımlı olduğu, ne zaman kimin hangi sorumlulukları alacağını ve hangi takımın canlıya çıkış sorumluluğunu üzerine alacağını netleştirdik. Karmaşık yapılarda bazen durup değişen koşulları baştan değerlendirmek ve ona göre yapılanmak gerekebiliyor. Bizim için de bu anlamda güzel bir deneyim oldu diye düşünüyorum.

Yalın ve Sade

Agile bakış açısı ile gelen önemli düşünce değişikliklerinden birisi de mükemmeliyetten uzaklaşmak ve evrimleşen çözümler üretmek. Bu nasıl işliyor?

Product Owner olarak en dikkat ettiğim noktalardan biri, "Bir de şu gelsin onu bekleyelim ondan sonra ikisini birlikte canlıya çıkalım" önerilerini, müşteri açısından sakıncalı bir durum yoksa elimine etmeye gayret etmek. Yılların getirdiği alışkanlıkla, "Her şeyi yapalım, mükemmel olsun, öyle müşteriye çıkalım" düşüncesi, özellikle böylesi değişim dönemlerinde ekiplerde ağır basabiliyor. Bu durum, ürünümüzü müşteri ile buluşturmada bizi geciktirdiği gibi, "Bir de şu gelsin öyle çıkalım" dediğimiz işin yapılması beklenmedik bir aksaklığa uğradığında, ürünümüz planladığımızdan çok daha sonra müşteri ile buluşmuş olabiliyor. Böyle durumlarda "Önce bu haliyle çıkalım, bir sonraki Sprint'te de diğer eklemeleri yaparak devam ederiz" yönlendirmesi yapıyorum ve takımın MVP kültüründen uzaklaşmamasına gayret ediyorum. Yaşanmış bir örnek olarak, müşteriye sunacağımız yenilikçi bir tarifeyi iyi bir dijital satın alma deneyimiyle birlikte çıkmak istemiştik. Aslında hem tarife, hem de dijital satın alma deneyimi, ayrı ayrı müşteriye güzel bir katma değer sağlıyordu. Birlikte çıkılması daha ideal bir senaryo olsa da bir zorunluluk değildi. Hatta konunun kapsamı büyüdüğü için müşterinin ürünle buluşmasını uzatabilecek riskler barındırıyordu. Ben burada iki katma değer yaratan parçayı birbirinden ayırarak önceliklendirme yaptım ve ürünün çok daha hızlı bir şekilde pazara çıkmasını sağladım. Böylelikle Squad'ın odağını dağıtmasını ve karmaşık senaryolar içinde boğuşurken yavaşlamasını da engellemiş oldum. MVP'yi canlıya hemen alarak müşteri reaksiyonunu hızlıca aldığımız için de üründeki bazı gelişim alanlarını müşterinin reaksiyonlarını gözleyerek fark edebildim. Bir sonraki adımda üründeki bazı planlanan noktaları değiştirerek pazara daha etkin haliyle yeni versiyonlarımızı sürebildik.

En küçük anlamlı parçaya odaklanmayı (MVP) seviyorum. Tüm detayları baştan öngörmek zorunda kalmamak ve test ederek ilerlemek, pazara hızlı bir şekilde çıkabilmek, aslında herkese keyifli geliyor. Bizim işimiz bire bir pazarlama. Bu yüzden, çok basit düşünmek kolay olmuyor. Genelde ortaya çıkarttığımız resmi daha sonra basitleştirmeye çalışmak gibi bir eğilimimiz oluyor. MVP kaliteden ödün vermek veya temel konularda eksiklerle pazara çıkmak demek değil aslında. Mesela, bir kez yaptığımız bir lansmanda Vodafone Yanımda uygulamamıza koyduğumuz görselde hata olduğunu fark ettik. Çok kısa bir süre sonra bu hatayı düzelttik ve herhangi bir şikâyet almadık. Ancak fark edilmeseydi riskli bir durum olabilirdi. Bu yaşanmış öğretiyle birlikte sonraki dönemde en ufak görsel ve metin değişikliklerini bile test etmeye başladık.

Bu pratikleri uygulamak başta çok kolay değildi. MVP kavramında olabildiğince basit düşünerek

ilerlemek hep işin kapsamını daraltmak olarak anlaşılabiliyor. Ancak, aslında adım adım ilerleyerek ürün ve hizmetlerinizi geliştirdiğinizde hem çıktı görme hızınızı artırıyor, hem de hata yaptığınız zaman bunu önden hızlı bir şekilde fark etmeniz ve aksiyon almanız kolaylaşıyor. Takım olarak başladığımız yeni bir pazaryeri projemizde MVP'mizi bir ürünün faturaya yansıtma yöntemiyle satışı olacak şekilde belirlemiştik. Ancak, daha önce ekipçe çalışarak canlıya çıkarmaya çalıştığımız dijital bir ürünümüz yoktu ve ürünlerin pazar beklentisini çok da gerçekçi analiz edemiyorduk. Bu noktada, MVP'mizi daha da küçülterek ürünlere talep toplama metodunu ilk MVP olarak belirledik. Bu aksiyon, hem daha önce deneyimlemediğimiz ortamları projede daha az risk varken denememize, hem de müşterilerimizden daha erken geri dönüşleri toplamamıza olanak sağladı. Bu anlamda, işi olabildiğince değer yaratan küçük parçalara bölmeyi ekipçe faydalı buluyoruz.

Agile eğitimlerimizin birinden bir kesit hatırlıyorum. Takıma şu soru soruldu: "Bu sabah Amerika'ya gideceksiniz. Sabah uyandınız. Evden çıkmak için neleri halletmiş olmalısınız?" Cevaplar bir sürü şeyi içeriyordu: "Kıyafetlerimi giyerim, dişimi fırçalarım, pasaportumu alırım, evi son bir kontrol ederim, vb." Sonra soru şuna döndü: "Bu sabah Amerika'ya gideceksiniz. Sabah uyandınız ve uçağınıza yetişmeniz çok riskli. Evden çıkmak için ne gerekiyor?". İnsanlar yalnızca cüzdanlarını, pasaportunu yanına alıp, pijamayla evden çıktılar. MVP böyle bir şey işte. Bir ürünü kullanıcıya sunabilmek için minimumda ne gerekiyor sorusunu gerçekten iyi yanıtlamak lazım. Takım olarak biz de bunu başarmakta ilk aşamalarda zorlandık. Doğal olarak, ürünün hayalimizdeki en iyi halini müşteriye sunmak istiyorduk. Öyle de yaptık. Sonra gördük ki, bizim hayalimiz ile kullanıcılarımızın beklentisi her zaman bire bir örtüşmüyor. Bu nedenle, önce küçük bir parçayı sunup, müşterilerin onu denemelerini sağlamalı, sonra kullanıcılarımızdan öğrendiklerimizle projeyi geliştirmeliyiz. Şu an tüm planlamalarımız bunun üzerine kurulu. Önce, yaptığımız işin kullanıcıya değer katacak en küçük parçasını sunuyoruz. Sonrasında, kullanıcıların davranışını takip ederek ve geribildirimlerini izleyerek bu parçayı geliştiriyoruz. Bu hem takım olarak bizi hızlandırıyor, hem de kullanıcıya onun beklediği nihai ürünü en doğru şekilde sunmamızı sağlıyor.

Beklentiyi doğru anlayıp, resmin bütününü kaybetmeden ihtiyacı adreslemek ve değerli küçük parçalarla ihtiyacı karşılamak gerçekten önemli. Bizim Squad'ımıza talepler genelde koca bir yumak halinde geliyor. Bu yüzden çoğu talepte ekibe ilk etapta neye ihtiyacımız olduğunu ve sonrasında gitmek istediğimiz yolu aktarmaya gayret ederek ilerlemeye çalışıyorum. Böylelikle MVP kapsamını ekibin katılımı ile beraber belirleyebiliyoruz. Mesela, son dönemde yaptığımız ve yeni canlıya çıkardığımız işlerden biri, en başta kocaman bir kapsam olarak gelmişti. Bu isteklerin hepsini beraber yapmaya kalkışsak en az 2 senelik bir çalışma gerekecekti. İki yıl gibi uzun bir süreden bahsettiğimiz için de aslında belki de bu sürenin sonunda bu projeye ihtiyaç dahi kalmayacaktı. Bu projeyi yapacak 3 Squad olarak bir araya geldik. Finansallar ve kullanıcı deneyimi anlamında olmazsa olmazları belirledik. Üzerinden hep birlikte tekrar tekrar geçtik ve en önemli olduğunu düşündüğümüz birkaç özellikte anlaştık. Böylece, işi 3 ay kadar kısa bir sürede müşterimize sunabildik. Şu an müşterilerin kullanımlarına ve deneyimlerine yönelik ihtiyaçlara göre diğer özellikleri ekleyerek ürünü geliştirmeye devam ediyoruz.

“Başarı” Nedir?

Senin için başarılı bir Squad olmak ne demek?

Başarılı bir Squad’ı düşününce aklımda, takımca ortak hedefe doğru hareket eden, karşılaşacağımız engellere de çözüm odaklı yaklaşabilen, Sprint içi görevlerini ekipçe birbirini destekleyerek tamamlayan ve bunları yaparken, müşteriye ve şirkete ne değer yarattım, diye sürekli sorgulayarak sonuç odaklı düşünen bir ekip canlanıyor. Bu noktada Product Owner’ın üzerine düşen önemli görevler, ekibe vizyonu doğru ve net aktarmak, öncelikleri doğru sıralama ile vermek ve işleri küçük ve yapılabilir parçalara bölmek. Aynı zamanda üründe yapılacak işlerin içeriği ve önceliklendirmesi ile ilgili Squad’dan gelen geribildirimleri dikkate almak da çok kritik oluyor.

Başarılı bir Squad olmak tek hedefe beraber koşmak demek. Her gün bu hedef için çalışırken yeni fikirler bulmak, daha iyisini yapmak, ekip arkadaşını geliştirmek ve ekip arkadaşından sürekli öğrenmek için heyecan duymak demek. Bence Product Owner’a düşenler; her gün daha fazla empati yapmak, ekipte denge noktası olmak, ekibe vizyonu ve stratejiyi çok iyi aktarabilmek, kalite ve başarı algısının takım olarak ortak bir dile dönüşmesini sağlamak, her gelişmeyi ve her kritik konuyu şeffaflıkla paylaşmak, önerilere açık olmak, buyurgan değil işbirlikçi olmak.

Squad’ın başarılı olması için hedeflerin tutturulması önemli. Bu başarıyı sağlamak ise sadece Product Owner’ın emeği ile mümkün değil. Tüm ekibin hedefleri anlaması ve görev bağımsız sahiplenmesi, tek hedefe koşması kritik. Bu noktada, Product Owner’ın Squad’ın hedeflerini takıma doğru aktarması ve ekibin aynı zamanda bir üyesi olması değerli. Product Owner olarak, ekibe şeffaf olmamız, ekiple aynı resme bakmamız ve ekip için fikir alışverişine açık bir ortam oluşmasına katkı sağlamamız çok değerli. Ekipteki kişiler fikirlerini açıkça ifade ettikçe ve bu fikirler hayata geçtikçe ürün daha fonksiyonel hale geliyor ve böylelikle başarı artıyor.

Benim için başarılı bir Squad, hedeflerini doğru şekilde koyan ve benimseyen, bu hedefler doğrultusundaki en iyi çözümü arayan, kendini sürekli iyileştirmeye ve geliştirmeye odaklanan bir ekip olabilmekten geçer. Bu noktada, Product Owner’lara önerim hedeflerinizin Squad’ınızdaki tüm rollerde bulunan arkadaşlar tarafından benimsenmesi sağlamaya çalışın. Başarı için “Bu işi neden yapıyoruz?” sorusunun cevabı herkeste net olmalıdır. Ekipteki herkes bu hedefin bir parçasıysa, her konuda fikrini dile getirebilmelidir. Herkesin kendi ana rolü dışındaki görev ve sorumluluklarda da söz hakkının olması ve değer verilmesi ekibi birbirine ve hedefe bağlayan en önemli etmen.

Takımın tamamının canlıya çıkardığımız her iş için kendini motive ve sorumlu hissettiğini görmek benim için başarılı olmak demek. Bence herkesin ortaya çıkardığımız başarıda kendi payını görebilmesi ve tüm ekip üyelerinin kendi fikirlerini de rahatça ortaya koyabilmesi başarıyı oluşturan unsurlar.

Bir Squad’ın başarısının temelinin, tüm takımın hangi hedefe hangi stratejiyle koştuğunu ve yol

haritasının ne olduğunu bilmesinden geçtiğini düşünüyorum. "Neyi neden yapıyoruz?" sorusunun cevabı herkeste net olmalı. Bu nedenle, Product Owner'ın takıma vizyonunu aktarabilmesi ve takımın bunu sahiplenmesi çok önemli. Daha da ötesi, stratejiyi ve yol haritasını oluştururken Product Owner takımın önerilerine kulak vermeli. Sistemi böyle kurduğunuzda, herkesin attığı adımı çok daha iyi sahiplenmesini, katkıda bulunmasını ve en önemlisi takım olarak hareket edebilmesini sağlıyorsunuz. Takım olarak hareket edebilmek ise Agile bir ekibin başarısının özünü oluşturuyor. İkinci önemli konu, hata yapmaktan korkmadan hareket edebilen bir organizasyona dönüşmek ve "test et, öğren" unsurunu gerçek anlamıyla hayatınıza sokabilmek. Bu konuda da Product Owner'lara büyük sorumluluk düşüyor. Siz ne kadar denemeye, risk almaya ve gerekirse hata yapıp öğrenmeye açık olursanız, ekibiniz de o kadar cesaretleniyor. Takımı bu konuda motive etmeyi unutmamak lazım. Bir Squad'ın başarısının ölçümü, kesinlikle Sprint'te planlanan işlerin yapılmasıyla değerlendirilebilecek kadar basit değil.

Product Owner'lara Tavsiyeler

- Agile iş yapış ile birlikte büyük bir değişim sürecinden geçtiğimizi düşünüyorum ve değişimin kendimizden başladığına inanıyorum. Bu nedenle, ben Product Owner'lara her zaman esnek düşünerek yeniliklere açık olmasını, doğru bildiği iş yapış şekillerini sürekli sorgulamasını ve kendini ve inandığı doğruları da yeniden değerlendirerek yenilemesini önerebilirim.
- Diğer Product Owner'lara önerim, vizyonları etrafında takımı da dahil ederek öncelikle net bir yol haritası yapmaları. Sonra, bu yol haritasını uygulamaya alırken uzmanlıklara saygı duymaları, ekipteki herkesi aynı derecede dinlemeye ve anlamaya gönüllü olmaları, vizyonu ve stratejiyi aktarmak için planlamaları beklememeleri, ekiple yakın çalışmaya ve ulaşılabilir olmaya özen göstermeleri. Gerektiğinde üstlerine iş almaktan, başarısızlıkları suçlamadan paylaşmaktan ve takım arkadaşlarına her koşulda inanmaktan çekinmesinler. Bunu yapınca, ekip de daha fazlasını yapmaya gönüllü oluyor. Bununla ilgili bir örnek paylaşmak isterim. Bizim Squad'ımız kurulduğunda ilk hedefimiz, mevcut yukarı yönlü satışı (upsell) hemen bir sonraki aydan itibaren %50 artırmak oldu. Bu hedef, başlangıçta ekibi epey zorladı. Bir araya geldik, onlara bu hedefin toplam şirket için ne ifade ettiğini ve neye hizmet ettiğini aktarmaya çalıştım. Olası bütün aksiyonları beraberce oluşturduk. Her aksiyonun bize ne kadar ek satış getirebileceğini ekip biliyordu. Bu yüzden, herkes kendi alanında fazlasını yapmaya gönüllü oldu. Gelişmeyi ve adetleri günlük takip ettik, Scrum tahtamıza yazdık, artışı gördükçe de beraber kutladık. Üç ayın sonunda da hedefimize hep birlikte ulaştık.
- Ürün İş Listesi yönetimini tek başınıza yapmaya çalışmayın. Ürün İş Listesinin tüm ekibin sorumluluğunda olduğunu, herkesin ona fikir ekleyebileceğini mutlaka tüm ekiple paylaşın. Herkesin katkısıyla en iyi çözümler ortaya çıkacaktır. Bu noktada, gerekirse Squad'daki herkes kendi fikir ve önerilerinin Product Owner tarafından öncelik alabilmesi için uğraşmalı. Bununla ilgili bir deneyimimi paylaşmak isterim. Bir planlama toplantısında ekipteki bir arkadaşımız DevOps ile ilgili bir talebi üzerinde ısrarla durmuştu. Ben, daha sonra ele alırız, diyerek diğer öncelikli işlere yönelsem de her yeni talebi

konuştuktan sonra kendisi tekrar tekrar kendi konusunun ne zaman konuşulacağını takip etti. Artık, rahatsız olmuştum. O Sprint için çok öncelikli konularımız vardı. Yine de kendisine söz vererek bu talebin ne olduğunu ve neden bu kadar gerekli olduğunu tekrar anlatmasını rica ettim. Canlıya alımlarda sıkıntı yaşandığını, hatta canlıda ciddi problemlerin yaşandığını anlattı. O hafta yaşanan 3 durumdan bahsetti ve benim için o anda bu talep en öncelikli iş halini aldı. Takımdaki arkadaşımızın bu ısrarı sayesinde, işlerimizi canlıya almadan önce birçok hatayı tespit edecek bir mekanizma geliştirmiş olduk. Bu mekanizma, sadece bizim Squad için değil, daha sonra diğer birçok Squad'ın da kullandığı bir yapıya evrildi.

- Eğer karakteriniz ve iş yapış şekliniz öyle şekillenmişse, sorumlusu olduğunuz ürünün/kanalın işlerinin detaylarına dalmamak için kendinizi çok zor tutuyorsunuz. Özellikle de geçmiş çalışma kültürünüzde tüm işlerin detayına hâkim olmanızın gerektiği bir noktadan geliyorsanız. Benim için işin "Ne yapacağız"ını belirleyip, "Nasıl yapacağız"ını ekibe bırakma süreci çok zor oldu. Tabii, benim için zor olduğu kadar ekip için de zor olmuştur eminim. Product Owner olduğum ilk zamanlarda, Sprint'e aldığım maddelerin nasıl yapılacağına dair detayları bile yazıyordum veya takıma detaylı anlatıyordum. İşleri takımdaki kişilerin kendilerinin sahiplenmesini beklemek yerine, iş atamaya çalışıyordum. Alışkanlık... Neyse ki bu girdaptan hızlı çıktım diyebilirim. Bu işe yeni başlayacaklara en büyük tavsiyem bu olur; bırakın, ekibiniz geliştirme maddelerini kendisi üzerine alsın ve nasıl yapılması gerektiğine dair önce kendi yolunu bulmaya çalışsın. Bu yolu bulduğu zamanki mutluluğu ve motivasyonu yaşasın. Gerekirse düşsün, kalkıp buradan öğrensin. İşin içinden çıkamadığı zaman gerektiğinde takım, gerektiğinde siz zaten ona destek olacaksınız.

- Product Owner, ekip üyelerine kendi sorumluluklarını alabilecekleri ortamı yaratmalı, iş taleplerini netleştirirken ekibin fikrini almadan ilerlememeli. Ortaya çıkan olumlu iş sonuçlarında başarıyı tüm ekip olarak paylaşmayı unutmayın.

- Geleceğin Product Owner'larına tavsiyem, takımınızı değerlendirirken, bakış açınızı Sprint'e alınan işlerin bitip bitmediğiyle, Burndown Chart'ların gidişiyle sınırlamayın. Başarının temelinde yatan ana nokta, iyi bir takım olabilmek. Bunu yaratmaya odaklanın. Gerisi zaten kendiliğinden gelecek.

- Mutlaka kendi ürününüze tüm detaylarıyla hâkim olun. MVP kültürünü içselleştirerek gitmek istenen ürün vizyonundaki anlamlı parçaları bulup işi mümkün olduğunca alt bileşenlere ayırmayı, müşteriyi sürekli dinleyerek ürünü evrimleştirmeyi alışkanlık haline getirin. Kararları kendi kişisel fikirlerinizle değil, veriye dayalı vermeyi temel alın.

- Tüm bu süreçte öğrendiğim bir gerçek var; Agile'da kişisel olarak başarısız olmazsınız, ekipçe başarısız olursunuz. Bu, aslında hem büyük bir sorumluluk, hem de büyük bir konfor alanıdır. Her bir başarıda ya da başarısızlıkta ekipteki herkesin doğrudan ya da dolaylı olarak katkısı vardır. Yüzdesel olarak en fazla katkıyı sağlayanı bulmak da sizi sonuca götürmez. Bunun nedenlerini iyi analiz etmek ve buradaki kazanımları hayata geçirmek, bir kez daha aynı hatanın yapılması riskini kaldırmanızı sağlar. O yüzden, takım olmaya ve takım olarak sürekli iyileştirmeye odaklanın.



Scrum Master Kimdir?

- **Scrum Master, ekibin daha odaklı ve verimli çalışması için en başta ekibin hayatını kolaylaştırmayı hedefleyen, onlara mümkün olduğunca zorluklardan arındırılmış bir çalışma ortamı yaratan ve Agile'ın daha iyi benimsenmesi ve uygulanması için aksiyonları alan, ekipteki Agile elçisidir.**

- Scrum Master rolü, tamamen sorumluluk duygusuyla ilgili bir rol. Ekibin yaptığı işlerin daha doğru bir düzlemde ilerlemesi için yardımcı olmak isteyen, gönüllü herkes Scrum Master olabilir.

- **Scrum Master, Squad içinde Agile ile hayatımıza yeni katılan ritüellerin yapıldığına ve Squad'ın tüm üyelerinin bu ritüelleri benimsediğine emin olan kişidir.**

- Scrum Master, sadece sorunu çözen değil, sorunu önceden görebilen, daha geniş perspektif ile düşünebilen, ekibi ve ekibin dinamiğini anlayan, onlara her türlü desteği veren ve onların daha iyisini üretebilmeleri için destek olan bir rol. Bazı yanlış anlamaların aksine, Scrum Master kesinlikle her türlü teknik konuda ekibe destek verebilecek bir ağabey/abla değil. Daha çok, yol gösterici ya da katalizör olarak hareket eden bir rol.

- **Scrum Master biraz gizli bir süper kahraman aslında. Ondan beklenenler, hem işlerin yolunda ilerlediğinden emin olması, hem takılan veya tıkanan konuları çözmesi, hem de ekibin motivasyonunun düşmesine izin vermemesi ve düşmesi durumunda da ekibin enerjisini yükseltecek aksiyonlar alması. Bunun için sahip olunması gereken en büyük beş özellik; enerji, empati yeteneği, gözlem yeteneği, yer aldığı ekipteki işlerle ilgili bilgisi ve sorumluluk sahibi olmasıdır.**

- Scrum Master, ekibin Agile çalışma sistematiğine ve pratiklerine alışma kısımlarında yol gösterici bir rol. Aynı zamanda tüm süreci gözlemleyen, iş yürütme kısmında ekibin önüne çıkan zorlukların aşılması için yardımcı olan, motivasyonu yüksek tutmaya çalışan ve ekip üyelerinin takım halinde çalışmasını sağlayan kişidir.

Bu Role Seçilme Hikayeleri

Scrum Master'larımıza sorduk, Scrum Master olmaya nasıl karar verdin ve seçildin?

Benim Scrum Master'lık hikayem biraz tesadüf oldu. Ekibimizdeki Scrum Master işten ayrılınca, bu göreve getirilecek, ekip dinamiklerini ve işi bilen birine ihtiyaç vardı. Önce tereddüt ettim, çünkü Scrum Master olmadan önce hep inandığım şey Scrum Master'ın bir yazılımcı olması gerektiğiydi. Sonuçta takımım, yazılım ürünü ortaya koyan bir takım ve Scrum Master'ın bu işten iyi anlaması gerektiğini düşünüyordum. Ben de teknik bir insanım, ancak, hiç yazılımcı olarak çalışmadım. Ben bu perspektiften bakarken, ekip arkadaşlarımın ve yöneticimizin tıkandığım konularda bana destek olacaklarını söylemesi ve beni bu yönde teşvik etmeleriyle birlikte bu role aday oldum. Gördüm ki rolün yazılımcı olmakla hiçbir alakası yokmuş.

Aslında, ekibime maksimum şekilde yarar sağlamak istediğim için Scrum Master olmak istedim. Ekip içi yaptığımız seçimde ekip arkadaşlarım da beni bu role uygun gördü.

Benim Scrum Master'lık sürecim aslında benden bağımsız olarak gerçekleşti. Ekibimizdeki Scrum Master'ın askere gitmesinden dolayı yeni bir Scrum Master ihtiyacı doğdu ve ekip arkadaşlarım bu görevi benim yapabileceğime inanmış olacaklar ki beni seçtiler.

Bence ne Agile geçmişinin ne de iş tanımının bir önemi var. Ben Scrum Master olmak istemedim aslında. Ekip içinde herkes kimin Scrum Master olmasını istediğini bir kâğıda yazdı ve bunları topladık. En çok benim ismim yazıldığı için "adaya" ben Scrum Master olarak devam ettim.

Ben Scrum Master özelliklerine sahip olduğumu düşündüğüm ve kendimi daha da geliştirebilmek için bu role aday oldum. Biz kısmen yeni bir ekibiz. İlk başladığımızda ekip üyelerinin bazılarını tanıyor, bazılarını ise tanımıyordum. İlk Sprint'imizdeki tanışma faslından sonra, ikinci iş olarak Scrum Master'ı seçmemiz gerekiyordu. Ben ve daha önceden tanıdığım bir arkadaşım aday olduk ve takıma vaatlerimizi sunduk. Diğer aday arkadaşım kek, börek gibi yemekler yapıp getireceğini vaat etti. Ben ise takımı şakalarımla güldüreceğimi söyledim. Toplantı boyunca vaatlerimiz havada uçup gitti. Sonuçta takım, adaylar arasında oylama yaptı ve iki aday berabere kaldı. Bu noktada yapılacak tek bir şey vardı; tabii ki yazı tura atmak. Parayı havaya fırlattık ve yere çarptığında sonuç belli oldu, Scrum Master bendim! Her ikimiz de bu rolü gerçekten istiyorduk ve eğlenceli bir şekilde, biraz da şakayla karışık olarak ilerledik. Sonuçta, ortada istek olduktan sonra bence kimin seçileceğinin de pek önemi yok.

Ben kendim Scrum Master olmak istemedim. Bu rol için konuşurken ekipteki herkes benim ismimi verdi. Böylelikle Scrum Master oldum. Agile'ın ne olduğunu bilmeden bir de Scrum Master rolünün ne olduğunu anlamaya çalıştım. Aldığımız Agile eğitimi sonrasında, sanırım herkesin en çok aklında kalan, Scrum Master'ın hizmetkâr olduğuydu. Sanıyorum, en beğenilen özelliği bu. İlk önce, ekibin Daily

Scrum'lara katılmasını sağlama, toplantı odaları tutma gibi bir rolümün olduğunu düşündüm (ekipçe düşündük). Zaman geçtikte, öneminin olmasının gerekliliğine inandık. Ekiptekiler, gerek Scrum Master olmam, gerekse bence şirketteki deneyimimden dolayı takıldıkları, sorun olan konularda bana geliyor. Böylelikle, hep birlikte engelleri aşmak için çalışıyoruz.

Yeni Rolünle Birlikte Hayatında Değişenler?

Daha önceki görevlerinle Scrum Master rolünü karşılaştırdığında, hayatında ne gibi farklar var?

Yaklaşık 6 aydır Scrum Master'lık yapıyorum ve başladığım ilk andan beri en yoğun hissettiğim duygu, sürekli kendimi ve ekibimi geliştirme ve iyileştirme isteği. Ekibime karşı ve paydaşlara karşı ciddi bir sorumluluk hissediyorum. Ekibimin duygu durumunu ya da iş yapışını etkileyebilecek herhangi bir durumda hemen aksiyon almaya çalışıyorum. Ekip ve birlikte çalıştığımız paydaşlar arasında zaman zaman bir işin aksama durumu ya da iletişim kopuklukları yaşayabiliyoruz. Bu tür durumlar olduğunda, hemen devreye girerek onları bir araya getirip birlikte çözüm üretmemizi tetiklemeye çalışıyorum. Bu açıdan baktığımda, kendimi ve ekibimi geliştirmenin yanı sıra, sanırım hayatımda artık birleştirici unsur olmak gibi yeni bir odağım var diyebilirim. Ayrıca, bir şeyleri değiştirdiğini, iyileştirdiğini ve ekibin daha mutlu çalıştığını görmek çok motive edici. Örneğin, Retrospektif'te ekip üyelerinin birbirlerine teşekkür edip hediye verdikleri bir yöntemi uyguladığımda, herkes kendisinin ve yaptığı işlerin takdir edildiğini bir kez daha gördüğü için motivasyonları yükseliyor. Ya da ekip üyeleri arasında küçük bir anlaşmazlık fark ettiğimde, konuyu büyütmeden bire bir olarak görüşmeleri için her iki tarafı da cesaretlendiriyorum ve çok basit bir yanlış anlaşılma durumunu karşılıklı iletişim kurarak kolayca çözdüklerinde bu gizli gerginliği fark ederek aksiyon aldığım için teşekkür ediyorlar. Fark yaratabilmek çok keyifli!

Agile çalışma şekli, aslında hiç bilmediğimiz, sıfırdan keşfettiğimiz bir şey değil. İnsanların günlük hayatına bile uygulanabilen bir bakış açısı olduğu için, ilk geçişte kendinizi sudan çıkmış balık gibi hissetmiyorsunuz. Ama, Scrum Master olmanın ayrı bir yönü var. Scrum'ın ne olduğunu öğrenmek, ekstra bir motivasyon yarattı bende. Scrum'la çizilen çerçevenin içini doldurmaya çalışmaktan zevk alıyorum. Ayrıca, normal hayatımızda bile uygulayabileceğimiz bir sistemi iş yerinde uygulayarak öğrenmek çok zevkli.

Scrum Master olduktan sonra aslında hayatıma yeni sorumluluklar eklendi. Kendimi ekipteki tüm işlerin nasıl yürüdüğünden ve daha da önemlisi nerede nasıl bir engel var, bunun için ne yapmak gerekir diye çok daha sorumlu hissetmeye başladım. Ekstra yapılması ve takip edilmesi gereken işler olduğundan, zaman

yönetimi kasımın geliştiğini düşünüyorum. Ayrıca, Scrum Master'lık kolaylaştırıcılık gibi temel bir sorumluluk getirdiği için liderlik yeteneklerinin de gelişmesine katkı sağlıyor.

Scrum Master olunca, kendimi birdenbire daha çok şey irdelerken buldum. Ekipten çok şey öğrendim. İlk başta ben bunlardan anlamam dediğim konularda, insanlara tavsiye verir oldum. Bireysellikten uzaklaşıp, takım ruhunun gerçekte ne demek olduğunu anladım.

Scrum Master olur olmaz ek sorumluluklar hissetmeye başladım. Her an ekibin motivasyonunu kollar hissediyordum kendimi. Ekibin birbirine hızla ısınması için birkaç aksiyonda bulundum. Bunun için en önemli yaptığım şey, ortak geçmiş oluşturmaya çalışmak, rutinin dışına çıkan somut hatıralar yaratmak oldu. Dışarıda yaptığımız ekip etkinliğine ek olarak bir ekip şarkısı yazdım. Sonra bu şarkıyı FreeZone Stüdyo'da kaydettik, videoya çektik ve tüm şirkete yaydık. Bu, bizim ekibimizin marka değerini şirket içinde artırdı. Bu yapıtın bir parçası olmak da tüm ekip üyelerinin ekibe ve birbirine aidiyetini artırdı.

Hayatımda ne değişti derseniz, ekibin yaptığı işlerin tüm bacaklarına bakmaya, resmi daha büyük görmeye başladım; hem Geliştirme Takımı'nın bir üyesi, hem de Scrum Master olmanın ne kadar zor olduğunu, tarafsızlığımı nasıl koruyabileceğimi, ekipteki kişileri oyuna nasıl dahil edebileceğimi öğrendim diyebilirim. Bu yeni rolle birlikte liderlik özelliklerimi geliştirdiğimi düşünüyorum. Ayrıca, rolümün getirdiği sorumlulukla birlikte Agile pratikleri daha iyi öğrendiğimi ve uygulayabildiğimi hissediyorum.

Bir Sprint'in Hikayesi

Scrum ritüellerimiz en önemli başlık. Scrum Master olarak bu ritüellerin en etkin şekilde yapılması için kolaylaştırıcı olmaya çalışıyoruz. Ekipler, Sprint'lerine[7], ilgili Sprint'te üzerinde çalışacakları işleri konuşarak, başka bir deyişle Sprint Planlama ile başlıyorlar. Sprint hedefimizi, öncelikli işlerimizi ve bunları nasıl yapacağımızı oluşturuyoruz. Sprint'in içinde her gün Daily Scrum'larda tüm ekibin katılımıyla 15 dakika içinde maksimum verim sağlamaya çalışarak güne başlıyoruz. Eğer Daily Scrum'da ekibin yaptığı paylaşımlarda daha detaylı konuşulması gerekliliği ortaya çıkıyorsa, Daily Scrum sonrasında ilgili ekip üyelerinin mutlaka bir araya gelmelerini tetiklemeye çalışıyoruz. Daily Scrum'larda, Scrum Master olarak, özellikle varsa, engeller ya da sıkıntılara odaklanmaya çalışıyoruz. Bu problemler gün içinde Scrum Master olarak bizlerin takıma yardımcı olmamız gereken olası işleri oluşturuyor[8]. Sprint'in geriye kalan zamanı, bu engellerin çözümü, ekibin enerjisini yüksek tutma ve mümkün olduğunca öğle yemeği organizasyonları ya da mesai sonrasında ekip etkinlikleri ile takımın iletişimini artırma odağında geçiyor. Sprint'in sonuna doğru ilerlersek, Sprint Review ile birlikte aldığımız işlerin doğurduğu sonuçlara odaklanıyoruz. Burada ekibin fikri önemli, işleri sadece bitti ya da bitmedi şeklinde ele almıyoruz. Biz bir işi yaptık yapmasına da, gerçekten yaptık mı? Gerçek bir iş çıktımız var mı, diye bakmaya çalışıyoruz. Ya da işi bitiremedik, ama "Sorun bir, neden bitiremedik" diye inceliyoruz. Bu ritüelin hemen arkasından Retrospektif geliyor. Retrospektif, aslında ekibin hâletiruhiyesine

[7]Genelde takımlarımız 2 haftalık Sprint'ler koşuyor.

[8]Scrum Tahta'sı üzerinde şeffaflığı ve odaklanmayı arttırmak için riskleri emojilerle belirleyebilir ve bir kişinin aynı anda üzerinde çalışabileceği iş sayısını sınırlayabilirsiniz.

göre yönlendirdiğimiz bir aktivite. Önemli olan, herkesin konuşmasını sağlamak[9]. Biz Agile çalışmaya ilk başladığımızda, ekipteki sinerjinin yürümesi için birkaç Retrospektif'in "kanlı" geçmesi gerek, derdik. Tabii ki gerçek kan değil. Mümkün olduğunca saygı çerçevesinde biraz tartışmak, karşındakini anlamak, yumuşak karnı nedir, neden hoşlanmaz, bilmek lazım. Bunu bilince, aslında hem iş yapış şeklini, hem de Retrospektif'lerin daha verimli geçmesini sağlayabiliyoruz. Bazı ekiplerde uyum geç olabiliyor. Özellikle bu uyumu sağlayana kadar Retrospektif'lerin sağlıklı, çıktısı olan ve o çıktıların hayata geçirilerek ilerlendiği ritüeller olması önemli[10]. Bunun için de her Retrospektif sonrasında aksiyon planlarımızı not almayı ve sonrasında da Sprint boyunca bu aksiyonları takip ederek gelişim alanlarımızı belirlemeye çalışıyoruz.

Genel olarak özetlemek gerekirse, Vodafone'da Scrum Master'lık odaklı bir rol olmadığı için genel amacımız, bir Scrum Master'ın yaptıklarını genel olarak takıma yaymak, herkesin süreci iyileştirmeyi ve üretkenliği tetiklemeyi sahiplenmesine zemin sağlamak. Bununla birlikte, tabii ki pürüzsüz bir Sprint için iş yapış biçimlerimizi kolaylaştırmaya çalışıyoruz . Sorularla ya da sorunlarla ilgili herkesin proaktif çözümler üretmesi için de yönlendirmeler yapıyoruz.

"Scrum Master takımın sekreteri değildir. Her şey ritüeller için toplantı odası bulmakla başladı. Bunlar Scrum Master'ın görevi sanılıyordu başta ya da kimse bu konularla uğraşmak istemiyordu. Retospektif'te durumu paylaşarak her Sprint'te bir kişinin seçilmesi ve yükün takım içinde dağıtılması kararlaştırıldı."

[9]Takımdaki sahiplenmeyi arttırmak için bazen Squad'daki farklı üyelerin Retrospektif'lere liderlik etmesini sağlıyoruz.

[10]"The Speedboard", "Facebook Reactions Retrospective", "Force Field Analysis", "Token of appreciation", "Mad, Sad, Glad" gibi farklı Retrospektif tekniklerini kullanıyoruz.

Etkileşimler ve Zorluklar

Scrum Master'lar olarak, aynı zamanda ekibin üyesi de olduğumuz için, zaten tüm ekip birlikte aynı masa etrafında oturuyoruz. Böylelikle gün içinde sürekli iletişim halindeyiz. Ekibin tüm üyelerine zaman ayırmak önemli. O yüzden her biriyle ayrı ayrı bağ kurmaya, onları gerçekten tanımaya çalışıyoruz. Gündelik yaptıklarını, nelerden hoşlandıklarını, karakter yapılarını, hobilerini öğrenmeye çalışıyoruz. Çoğu zaman iş dışı da birlikte zaman geçirdiğimiz oluyor. Çalışırken aynı zamanda eğlenmeyi başarabiliyoruz.

Genelde Product Owner'larımız da yine bizimle aynı masada oturduğu için onlar da hep ekibin içinde ve tüm etkinliklerimizin bir parçası. Benzer bir şekilde, Tribe Lead'lerimiz de Squad'lara yakın olarak çalışıyorlar. Her gün aynı çalışma ortamını paylaştığımız için gün içinde her konuyu kendileriyle görüşebiliyoruz. Özellikle çözüm sağlayamadığımız konularda onların desteğini her zaman alabiliyoruz.

Ayrıca, diğer Squad'ların Scrum Master'larıyla da ayda bir kere bir araya gelerek birbirimizle deneyimlerimizi ve ekiplerimizde iyi giden ya da çözmemiz gereken konuları paylaşıyoruz ve birbirimizden yeni şeyler öğreniyoruz. Agile'ın hayatımıza yeni getirdiği araçlar ya da işleyişler hakkında birbirimize yardımcı oluyoruz.

"Eskiden Sprint Planlama sonrasında ekiple birlikte Scrum Tahta'mızı hazırlamaya çalışıyorduk, ancak, dağılan ekibi tekrar bunu hazırlamak için bir araya getirmek kolay olmuyordu. Scrum Master etkinliklerinden birinde, bir başka ekibin Sprint Planlama

sırasında bir yandan ekipçe birlikte Scrum Tahtası'nı da hazırladıklarını öğrendim. Biz de bunu uygulamaya başladık ve böylece ekip dağılmadan hep birlikte tüm hazırlıkları halletmiş oluyoruz."

Tribe içinde Agile yapının işleyişi özelinde tüm Tribe'daki Scrum Master'lar ve ilgili bütün Chapter Lead'ler zaman zaman bir araya geliyoruz. Ekiplerdeki kişiler, yaşadıkları zorluklar, süreçler ve beklentiler çerçevesinde tüm Squad'lardaki genel gidişatı değerlendirerek iyileştirme aksiyonlarını hep birlikte hayata geçirmeyi tartışıyoruz.

Scrum Master'lık, hayatımıza Agile dönüşümü ile birlikte girdi. Tüm şirket olarak, Scrum Master'lar da dahil bu rolün görevini henüz keşfetme aşamasındayız. Farklı pratikler deneyerek ekibimiz için en uygun yöntemi bulmaya çalışıyoruz. Yönetim de bu konuda esnek ve destekleyici bir tutumda. Onlar da hem Agile pratiklerini, hem de bu pratiklerin uygulanmasında Scrum Master'ın rolünü gözlemliyorlar. Bu role ve Agile çalışma şekline alışmaya çalışırken önümüze çeşitli engeller de çıkıyor elbette. Bunlara kısaca bakarsak:

- Normalde Scrum Master'lık tam zamanlı bir rol, ama bizde Squad'ın üyelerinden birisi bu görevi de ek olarak üstleniyor. Bu yüzden, zorlayıcı bir durum. Scrum Master olarak aynı zamanda Sprint'in içinde görev alınca, işin hengamesine kapılabiliyor ve bazı iyileştirilmesi gereken şeyleri göremeyebiliyorsunuz. Hatta, gerektiğinde, tarafsız davranamayabiliyorsunuz.

- Şirketteki birçok kişi için bu yeni bir rol. O yüzden, hepimiz bu rolü kendimiz keşfetmek durumundayız. Yaşayarak öğreniyoruz. Hem keyifli hem de zor bir durum.

- Farklı yetkinlikleri olan ve farklı deneyimlerden gelen kişilerden oluşan yeni bir yapı Agile takımlar. Haliyle, burada da ekip içinde ilk başlarda iletişim zorlukları ve takım olmada problemler yaşanabiliyor.

- Büyük bir şirkette değişim her konuda kolay ya da hızlı olamayabiliyor. Şirket içindeki işleyişle alakalı bazı sorunların uzun süreler boyunca devam etmesi ve Sprint'lerde aynı sorunlardan tekrar tekrar söz etmek, bunları yeterince ilerletememek, motivasyon kaybına neden olabiliyor.
- Tabii, olmazsa olmaz bir sorun; toplantı odası bulmak.

- Biz iki haftalık Sprint'ler koşuyoruz. Bazı takımlarda bu iki hafta içindeki iş dağılımı istendiği gibi ilerleyemeyebiliyor. Sprint'in ilk haftası daha yavaş, ikinci haftası ise yoğun geçebiliyor. Bu alçalan ve yükselen ruh hali, ekiplerde stres yaratabiliyor.

Scrum Master'ın Yakın Gelecek Gündemi

Scrum Master olarak önümüzdeki günlerde seni ve takımını neler bekliyor?

Scrum Master olarak, kısa ve orta vadede özellikle odaklandığım ana konular, takım oyununu geliştirmek ve takımın önündeki

engelleri ortadan kaldırmak. Bu anlamda Retrospektif'lere eğilmeyi, burada farklı pratikleri denemeyi planlıyorum.

Ekibimiz 8 kişiydi, aramızdan ayrılanlar oldu. Ayrılan arkadaşlarımızın yerine yenilerinin gelmesi zaman alacağından ekibin motivasyonunun etkilenmemesi için ciddiyeti elimden bırakmıyorum. Durum gereği, giden arkadaşların ekipteki rollerini üstlenmem gerekti. Takım demek de sanırım bu; gerektiğinde fazla sorumluluk almak ve takıma destek olmak. Orta vadedeki en büyük görevim ise yeni arkadaşlar geldiğinde, tekrardan bir ekip ruhu yaratmak olacak.

Scrum ile ilgili daha çok okuma yapmak, öğrenmek, ilk amacım. Gerçekçi olmak gerekirse, layıkıyla bunu yapamadığımı düşünüyorum. Artık, gerekirse, takım içindeki işlerimi biraz askıya alarak kendi donanımımı geliştirmeye daha fazla eğileceğim.

Biz ekip olarak hem dijital hem de fiziksel tahta kullanıyoruz. Fakat fiziksel tahtayı çok etkili bir şekilde kullanmayı henüz gerçekleştiremedik. Bu nedenle, kısa vadede fiziksel tahtayı daha eğlenceli ve anlaşılır hale getirmek, odaklandığım noktalardan. Uzun vadede ise daha otonom bir yapıya gidebilmek için çalışmayı planlıyorum. Uzun yıllardır çalışan ve belirli bir çalışma prensibini benimsemiş kişiler için Agile'a geçmek o kadar kolay olmuyor. Örneğin, ekip, sabah toplantılarını takımın birbiriyle senkronize olduğu bir pratik yerine, Product Owner'a rapor verilen bir ritüel olarak görebiliyor. Ekiptekilerin hem Agile bakış açısını hem de iş yetkinliklerini geliştirerek, otonomiyi artırmayı hedefliyorum.

En önemli şey, ekip motivasyonu. Ekibin yaptığı işleri daha görünür hale getirip, aslında yaptıkları işlerin ne kadar önemli olduğunun herkes tarafından görülmesini sağlamak için şeffaflığı artırmayı hedefliyorum. Ayrıca, kendi rolümle ilgili daha çok okumaya ve öğrenmeye yatırım yapmayı planlıyorum. Global Scrum Master Sertifikası almak gibi bir hedefim var.

Kısa vadede Sprint ritüellerinin daha etkin geçmesi için geliştirme yolları bulmaya devam edeceğim. Çünkü ekibin uzun zaman alan toplantılarda sıkılmaları motivasyonlarını dağıtabiliyor. Bunu engellemek için tüm ekip üyelerinin aynı oranda rol alacağı şekilde ritüelleri gerçekleştirmeye çalışıyorum. Orta vadede ise ekibin T-Shaped bir yapıda çalışması, başka bir deyişle herkesin farklı uzmanlık alanlarını öğrenip o alanlarda kendilerini gerçekleştirmeleri için aksiyonlar almayı planlıyorum. Örneğin, ekipteki herkesin yine ekip içinde farklı bir konuda uzman kişiyle "eş" olmasını ve işleri birlikte yürütmelerini, böylece mümkün olan seviyede birbirlerinin işlerini öğrenmelerini sağlamayı istiyorum. Böylece, bir ekip üyesinin öngörülemeyen bir işi çıktığı zaman, bu durumdan daha az etkilenerek ilerleyebiliriz.

Scrum Master'lara Tavsiyeler

- Her insanın farklı bir kapasitesi, her ekibin farklı bir dinamiği var. Scrum Master'lara bir terzi gibi hareket etmelerini, ezbere pratiklerle değil, ihtiyacı anlayarak yaklaşımda bulunmalarını tavsiye ederim.
- Ekip ruhu yaratmak için elinizden geleni yapın. Tek hedef, iyi işleyen bir takım yaratabilmek!

- İş hedefleri ve bu anlamdaki ölçümlemeler dışında temel bazı Scrum ölçüm araçlarını kullanabilirsiniz; Burndown Chart, Sprint Hızı, Sprint'e alınan ve tamamlanan iş oranı gibi. Ancak, bunlar sadece araçlar, çok anlam yüklememek gerekir. En anlamlı metrik ise motivasyon. Her gün ekip arkadaşlarınızın yüzlerine bakın. Ekipteki arkadaşlarınız birbirine gülümsüyorsa ve eğlenerek çalışıyorsa, gerisi gelir.

- Geribildirim kültürü çok önemli. Buradaki eksiklik, Retrospektif'leri de ciddi anlamda etkileyebiliyor. Tüm takım olarak geribildirim ve etkin iletişim eğitimleri alarak başlayın.

- Nasıl ki ben kralım diyen kral değilse, ben Scrum Master'ım diyen kişi de Scrum Master olamamıştır. Öte yandan, "İyi hacker'ları herkes bilir ama en iyisini hiç kimse" sözünde olduğu gibi, Scrum Master da Scrum Master'lığını hissettirmemelidir diye düşünüyorum. Bu sebeple, bu zorlayıcı rol, ister istemez görünmez bir el gibi hareket etmek zorunda kalıyor. Bu durum, sizi demotive etmemeli. Herkes mutluysa ve başarılı takımlar varsa, bilin ki siz işinizi iyi yaptınız.

- Retrospektif, takımın iyileşmesi için en önemli ritüel! Bu yüzden, farklı teknikler kullanın.

"Sprint'lerimizin Retrospektif ritüellerinde yorum yapmaya çekiniyor ya da yapmak istemiyorduk. Bu nedenle Retrospektif'lerimiz kısa ve verimsiz geçiyordu. Bu ritüeli daha verimli kullanabilmek adına Sli.do diye anonim olarak yorum yazabileceğimiz bir araç kullanmaya başladık. Tüm ekip olarak Sprint boyunca Retrospektif'te konuşulması gerektiğini düşündüğümüz konuları buraya yazıyoruz. İsteyen kişiler diğer ekip arkadaşlarının yazdıklarını oylayabiliyor. Sonrasında da Retrospektif'lerde yüksek oy almış konulara odaklanıp Sprint için aksiyonları çıkarıyoruz."

- Scrum Master'lar, Agile dönüşümün elçileri aslında. Bu anlamda, şirket genelinde iyileştirmelerde fikirlerinin alınmasının ve süreçlere dahil edilmelerinin önemli olduğunu düşünüyorum. Bu doğrultuda, sadece takımınızla sınırlı kalmayın, aktif bir şekilde organizasyonun dönüşümünün içinde yer alın.

- Ekibin / kendinin gelişimi için, diğer ekiplerin deneyimlerinden mutlaka yararlanılmalı. Denenmiş ve başarıya ulaşılmış hikâyeleri uygulamaya çalışın.

- Mümkün olduğunca ekipteki her rolü biraz tecrübe etmeye çalışın. Böylelikle her rolün gerekliliğini ve zorluklarını daha iyi anlayabilir ve empati kurabilirsiniz.

- İyi bir Scrum takımı tamamen kimya işi. Takım üyelerinin birbirini iyi anladığı ve sorumluluğu paylaştıkları dünyada Scrum çok zevkli. Scrum Master'ların da buradaki rolü diğer takım üyelerinden farksız: takım oyununu benimsemek ve benimsenmesi için herkesi cesaretlendirmek.

- Öncelikle, Scrum Master'ı olduğunuz ekibin üyelerini tek tek tanıyın: neyi severler, neye gücenirler, neye gülerler, neden sinirlenirler? Sonra, her birinin sorusuna/sorununa ona özel çözümlerle yaklaşın. Ritüellerin ne zaman nerede yapılacağı, fiziksel veya dijital tahta oluşturma gibi kararları, hep ekipçe oy çokluğuyla alarak ilerleyin. Her konuda tüm ekibin katılımı önemli. Her zaman birlikte başarmayı hedefleyin.


vodafone
Maslak
Gelecek heyecan verici.
Hazır mısın?
Gülümseten Teknoloji
Yanınızdayız

Chapter Lead Kimdir?

- **Chapter, Agile organizasyonda Tribe yapısı içinde benzer uzmanlık ve becerilere sahip kişilerin oluşturduğu ortak bir sanal takım yapısıdır. Burada amaç, benzer uzmanlıktaki kişilerin ekiplerinde yaptığı iş ile ilgili paralelliği sağlayabilme ve kişisel gelişim noktasında birbiriyle iletişimde kalmalarını sağlayacak bir yapılanma kurabilmektir. Aynı zamanda ilgili uzmanlık alanı kapsamında Squad'lar üstü bir hizalanma görevi de almaktadır.**

  Chapter Lead ise bu Chapter'ın hedefleri doğrultusunda üyelerine liderlik eden, Chapter üyelerinin kariyer/gelişim süreçlerinde onlara koçluk yapan kişidir.

- Chapter Lead, en özet haliyle, lideri olduğu Chapter seviyesinde uzmanlık bilgisine sahip, Chapter'ındaki çalışanların potansiyelleri ve performanslarının yönetiminden, yetkinliklerin gelişiminden sorumlu olan, Chapter içinde bilgi ve beceri paylaşımlarının artırılması için gerekli ortamı sağlayan, ekibini dinleyen, varsa eğer zorlukların veya problemlerin çözümünde yol gösteren ve ekibine koçluk yapan kişidir.

- **Chapter Lead, farklı Squad ekipleri içinde benzer işleri yapan arkadaşlar arasında doğru iş yapma pratiklerini ve senkronizasyonu sağlayan kişidir.**

- Chapter Lead, farklı masalarda çalışan aynı sorumluluk alanına sahip çalışanların işlerinin kalitesinden, bilgi/becerisinden ve koordinasyonundan sorumlu kişidir. Aynı zamanda çalışanların motivasyon noktasıdır. Chapter Lead, insanlardan sorumlu olduğu kadar, farklı ekiplerin birbirleriyle koordineli bir şekilde çalışmasından da sorumludur. İş sorumluluklarının Squad'lar üzerine dağılması, aynı sorumluluk alanına sahip ekiplerin birbirinden uzaklaşmasına sebep olabilir. Bu noktada, ekipler arasındaki ortak aklı Chapter Lead sağlar.

Yeni Hayatla Birlikte Değişenler

Chapter Lead'lerimize sorduk, bir önceki görevin ve yeni görevin arasında ne gibi farklar var?

Chapter Lead olduktan sonra yönetim stilimde ciddi değişiklikler olduğunu söylemek zorundayım. Temelde daha geleneksel, delegasyon ve destek bazlı bir yönetim tarzından, daha çok koçluğun merkezde olduğu, daha çok dinleyen, empati kuran, bireysel beceri ve yetkinlikleri geliştirmeye ve büyütmeye odaklandığım bir liderlik yaklaşımına evriliyorum. Tabii, bizler Chapter Lead olarak, aynı zamanda aktif olarak bir Squad'ın içinde de takım üyesi olarak görev alıyoruz. Bu anlamda, hem Chapter'ımızla hem de Squad'ımızla ilgili odağımız var. Taktığım Squad üyeliği şapkasıyla taşın altına bizzat elimi sokma ve uzmanlık alanımla ilgili doğrudan ilk kaynaktan iç görüler elde edebilme fırsatını da yakaladığımı belirtmek isterim.

Chapter Lead olduktan sonra belirsizliklerle yaşamayı ve çözüm üretmek için kendimi motive etmeyi öğrendim. Bu belki de Agile yapıya geçtikten sonra birçok rolün yaşadığı bir süreç. Agile yapı, bu belirsizliklerden bir nevi besleniyor ve bununla yaratıcılığı tetiklemeye çalışıyor. Eski yapıdaki gibi "Bu benim işim, şu senin işin" gibi keskin ayrımlar yok. Onun yerine, herkesin üretilen değeri sahiplendiği ve yüksek iş birlikteliği ile çözümün bir parçası olmayı öğrendiği bir yapıdayız. Bu da bazı gri noktaları beraberinde getiriyor elbette. Her ne kadar bu gri noktalar ilk başta yorucu olsa da, iş birlikteliğini öğrendikçe bundan motive olmayı da öğreniyorsunuz.

Chapter Lead'lik ekiple aramdaki çalışma yöntemini ciddi seviyede değiştirdi. Eskiden ekip içinde yapılacak işleri ve çalışmaları kendi içimizde planlayıp aksiyona geçirirken, artık, bu işlerin Squad ekipleri içinde planlanıp yapılmasını sağlamaya ve işleri bu akış üzerinden takip etmeye çalışıyoruz. Bu, özellikle organizasyonun bir kısmı henüz Agile çalışmayan bir yapıda olduğundan, ayrı bir zorluk yaratıyor. Organizasyonun bir kısmı beni hâlâ "müdür" olarak tanımlayıp işleri bu kapsamda takip etmek istiyor. Ben ise onlara "Hayır ben Chapter Lead'im, bu cevaplanmasını istediğiniz soruları ya da yapılması gereken işleri ekibe iletmeniz gerekir" diyorum. Bu da insanı tabii biraz yoruyor.

Chapter Lead olmamla birlikte, kendi Squad'ım içinde aldığım yapay zekâ tabanlı dijital asistanımız Tobi'nin ürün geliştirme ve yapay zekâ eğitim süreçlerindeki işlerime ek olarak bu görevi üstlenmeye başladım. İkisi içindeki dengeyi kurma ve Chapter hedefleri için de çalışmalarımı sürdürüyorum. Yeni olan, sanırım, insan odağı. Uzmanlık alanım haricinde, kişilerin yetenek gelişimini desteklemeye de odaklanmak zorundayım.

Ortalama Bir Haftanın Hikayesi

Bir Chapter Lead'ın haftalık zamanı, yarı yarıya dahil olduğu Squad içindeki işler ve Chapter Lead rolündeki sorumlulukları çerçevesinde geçiyor diyebiliriz. Bir taraftan, spesifik bir Squad içinde görev alıp tüm Agile ritüellere ve Squad bazlı toplantı ve etkinliklere dahil olurken, diğer taraftan, Chapter'daki üyelere zaman ayırmaya çalışıyoruz. Liderliğini yaptığımız uzmanlık alanlarıyla ilgili bilgi ve deneyim paylaşımını artırmaya yönelik önceden planlanan veya spontane gelişen platformlarda toplu (Chapter Huddle) veya bireysel bir araya gelerek Chapter içi yetkinliklerin gelişmesine destek oluyor ve en temel seviyede Chapter'larımızdaki insanların "mutlu" ve "değer yaratan" bir çalışma haftası geçirdiğinden emin olmaya çalışıyoruz.

Her Chapter, kendi üyeleri ile birlikte yol haritası üzerinde çalışıyor. Burada farklı kanallardan müşteriye ulaştırdığımız ürünlerden çıkan öğrenimlerin Chapter içinde paylaşılması, global ölçekte bizim oyun alanımızı etkileyebilecek teknoloji ve konseptleri takip etme ve bu çıktıları iş hedeflerimizle nasıl örtüştürüp müşterimize daha iyi bir hizmet sunabiliriz gibi konuları Squad üstü bir yaklaşımla Chapter içinde ele alıyoruz.

Chapter'daki arkadaşlarımızla yüz yüze görüşmeler yaparak işler özelindeki soruları yanıtlamaya çalışıyoruz. Eğer yardım talebi gelirse, ekip arkadaşlarımızla birlikte işin detayına girip çözüm için birlikte çabaladığımız da oluyor. Bir yandan da Chapter'ı daha etkin çalışır hale getirmekle ilgili süreçler üzerinde çalışmaya ve aynı zamanda ekibimize dahil olacak aday arkadaşlarla iş görüşmeleri yapmaya devam ediyoruz. Sürdürülebilir bir yapı kurabilmek ve kültür dönüşümünün itici gücü olmak, ana odağımız diyebiliriz. Ayrıca, mümkün olduğunca, başka Squad'ların Sprint Review'larına aktif bir şekilde katılmaya çalışıyoruz. Böylelikle, farklı Squad'larda ne gibi gelişmeler var, görebiliyor ve kendi Chapter'larımızla ilgili olarak gelecek planlarımızı yönlendirebiliyoruz.

Paralelde, mutlak suretle, diğer Chapter Lead ve Product Owner arkadaşlarla ve Tribe Lead ile bir araya gelerek birbirimizi her türlü konuda (çalışanlar için geribildirim, iş öncelikleri gibi) beslemeye çalışıyoruz. Yine liderlik ettiğimiz spesifik uzmanlık alanı ile ilgili Tribe dahilinde tüm Squad'lar açısından gelişme kaydetmemiz gereken ve yine diğer Tribe veya Agile çalışmayan ekiplerle birlikte ilerletilmesi gereken işler varsa, bu konulara odaklanıp inisiyatif alıyor, daha iyi performans göstermemiz için gerekli aksiyonların ilerletilmesini sağlamaya odaklanıyoruz. Bir Chapter'ın yaşanmış örneğini paylaşacak olursak, yakın bir zaman önce, yabancı müşterilerin, Bilgi Teknolojileri ve İletişim Kurumu zorunluluğu gereği, belirli bir tarihe kadar sisteme yabancı kimlik numaralarının tanımlanması gerekiyordu. Bu müşterileri aramak için ayırdığımız zaman ve efor, tüm faturalı Squad'lara ait aktivitelerde satış kaybına yol açmıştı. Bu aktiviteler nedeniyle ayı eksik kapatmamak için inisiyatif alarak iş ortağı yönetimi ekipleriyle hızlıca bir araya geldik, aksiyon planı çıkardık ve sahada tele-satış uygulanmasını sağlayarak aynı ay içinde tüm Squad'lar özelinde kaybettiğimiz satış adedine hızlıca geri ulaşmayı başardık.

Bir Chapter Nasıl Çalışır?

Chapter olarak bir araya gelmeye "Chapter Huddle" ismini veriyoruz. Ayda en az bir defa bir araya gelmeye çalışıyoruz. Bazen bu sayı iki olabiliyor. Ancak, genelde ikincisinde iş konuşmak yerine, birlikte zaman geçirerek ekibin kaynaşmasını tetiklemeye çalışıyoruz. Tabii, Chapter'ın iletişimi için illa bu toplantıları beklemeye gerek yok. İletişimimizi sürekli ve yakın, ihtiyaçlar dahilinde paylaşımlarımızı aktif tutmaya çalışıyoruz.

"Chapter Huddle" İçeriği Nasıl Olur?

İçerik, Chapter bazlı, spesifik ihtiyaçlara göre değişebiliyor. Genelde, bir araya gelmeden önce, tüm üyeler ilgili toplantının gündem maddelerini önceden belirlemiş oluyor. Toplantı da bu gündem takip edilerek ilerletiliyor. Ancak, genel olarak maddelersek:

- Chapter'ın kendi yol haritasının gözden geçirilmesi ve takibi,
- Her Chapter üyesinin, kendi Squad'ı özelinde bir önceki ayı performans bazında nasıl kapattıklarını, varsa örnek olabilecek iyi uygulamaları, çıkardıkları öğrenimleri ve bir sonraki aydaki öncelikleri ve planları paylaşması,
- Doğru pratiklerin paylaşımı ve yeni teknolojiler üzerine fikir alışverişi,
- Yeni bir konu seçip birlikte araştırma ve öğrenme, böylelikle hep birlikte kendi yetkinliğimizi artırmaya yönelik fırsat oluşturma,
- Retrospektif, hatalarımızdan öğrenme ve iyileştirme aksiyonlarını belirleme

gibi konularımız olduğunu söyleyebiliriz. Bu bir araya gelmeler, tüm üyelerin birbirlerine geribildirim verebilecekleri çok sağlıklı ve şeffaf bir platform yaratıyor. Chapter Lead olarak, bizim bu toplantıdaki amacımız, Chapter içindeki bilgi ve deneyim paylaşımına zemin hazırlamak, konuşulan konular çerçevesinde varsa katkıda bulunabileceğimiz noktalar ile ilgili yorumlarımızı vermek ve hem kişilerin hem de iş performansının gelişimine katkıda bulunmak oluyor. "Chapter Huddle" platformu, aynı zamanda Chapter'daki kişilerin rutin hayatlarına döndüklerinde, birbirleriyle olan iletişime ve etkileşime de katkıda bulunuyor.

Chapter Lead'in ve Üyelerinin Yetkinlik Gelişimi

Chapter Lead olarak kendini geliştirmek için neler yapıyorsun?

Chapter Lead rolü, hem Türkiye hem de Vodafone özelinde yeni sayılabilecek bir rol. Her rolde olduğu gibi, gelişimin büyük bir kısmı, işin içine girerek, inisiyatif alarak ve daha fazla deneyimleyerek geliyor. Şirket, benim de dahil olduğum, gerek Agile çalışma şekli, gerekse koçluk yetkinlik gelişimi, geribildirim becerileri gibi birçok seviyede faydalandığım spesifik eğitim programları düzenliyor. Ancak, bu roldeki gelişimimin görece en büyük kısmını deneyimleyerek, ekibe ve uzmanı olduğum alana daha fazla zaman ayırarak elde ettiğimi düşünüyorum.

Chapter'daki kişilerin gelişiminin, kişilerin hem kendilerini geleceğe hazırlamaları, hem de birlikte çalıştıkları Squad'ların performanslarına katkı noktasında oldukça kritik olduğunu düşünüyorum. Yeni başlayan bir çalışanın gelişiminde mentörlük yaparak ve "eşleştirme sistemi"nin uygulamasının hayata geçirilmesini sağlamak vasıtasıyla verim alırken, profesyonel seviyede bir çalışanın gelişiminde ise koçluk yaklaşımından daha çok verim alabiliyorum. Yine şirket içi ve çevrimiçi eğitimler, destek aldığımız kaynaklar.

Açıkçası, rolümle ilgili yaşayarak öğrenme en önemli pratiğimiz oluyor. Sonuçta, Agile çalışma şekli, şirketin bütününde yeni sayılabilecek bir konu. Bunun bütün paydaşları olarak, genel bir deneyimleme ve öğrenme rutinindeyiz. Bu noktada, kendi adıma en çok uyguladığım davranış ise otokontrol oluyor, sanırım. Mümkün mertebe, eski tarz "müdür" davranışlarından ve pratiklerinden uzak durmaya çalışıyorum. Bununla birlikte, şirket içinde her alanda kendini geliştirmek için sayısız fırsat ve mekanizma mevcut. Ancak, sanırım şirketimizin genel anlamdaki sıkıntılarından biri mevcut iş temposundan başını kaldırıp bu konulara zaman ayırabilmek. Ben de zaman ayırabildiğim kadarıyla şirketin eğitim fırsatlarını değerlendiriyorum. Bunun yanında, konumla ilgili şirket dışındaki toplulukları ve etkinlikleri takip etmeye çalışıyorum.

Chapter toplantılarımızın gündemini, herkesin yeni bir şeyler öğrenebileceği, kendi farklı pratiklerimizi anlattığımız bir yapıya çevirdik. Öğrenme işini sürekli hale getirmeye çalışıyoruz; yaşayan ve birlikte öğrenen bir organizasyon olmayı hedefliyoruz. Bunun yanında, ekipteki arkadaşlarımızla bire bir takip ettiğimiz bir yetkinlik gelişim planımız da mevcut. Bu plana göre, kişilerin yetkinlik yelpazeleri ile ekibin ihtiyaç duyduğu yetkinlikleri beraber değerlendirip karşılıklı faydayı merkezinde tutan bir yapı ile bu gelişimi izliyor ve destekliyoruz.

Geçiş aşamasında olduğumuz için, tüm süreçlerde anlaşılır bir belirsizlik var. Bu rolde

çalışan başka bir örnek bulamadığım için, neyi nasıl yapacağıma kendim karar veriyorum ve sorumluluk alanımı kendim çiziyorum. Eğitimler sağlanıyor, ancak, bu eğitimde alınan bilgileri tam olarak uygulamak çok zor. Aslında, deneyip yanılarak ve sürekli iyileştirerek kendi yöntemlerimizi geliştiriyoruz.

Kişilerin beceri gelişimi içinse, kendi uzmanlık alanımdaki konularla ilgili ekip içi periyodik eğitimler düzenliyoruz. Adobe Analytics, Target, Inapp ve Google Firebase gibi uygulamalar için teknik ekiplerden eğitimler alarak kendimizi geliştiriyoruz. Bunun yanında, dijital pazarlama, büyük veri, e-ticaret alanındaki gelişimimiz için dışarıdan firmalar ile bir araya gelerek iş geliştirme aktivitelerinde bulunuyoruz.

Kendimi online eğitimler ve şirket içinde yaptığım paylaşımlar ile geliştirmeye çalışıyorum. Şirketin bu konuda eğitimlerine de katılmaya başladım. Özellikle liderlik, koçluk, motivasyon ve geribildirim konularında kendimi geliştirmeye devam ediyorum. Bu bir süreç olduğundan, doğru adımları atmak için elimden geleni yapıyorum. Rolün bizimle birlikte anlam kazanacağının farkında olarak yeni bir yapılanma kurabilmek, çok önem verdiğim bir konu.

Chapter'ımdaki üyelerin gelişimi için ise, şu an bire bir toplantılarla eksik olan noktaları birlikte konuşuyoruz. Bir sonraki aşamada bunlarla ilgili aksiyonları almaya başlayacağız. Burada sürecin yönetimi konusunda yardımcı olabilecek koçluk pratiklerinin (GROW gibi) kullanımının yararlı olduğunu düşünüyorum.

Etkileşimler ve Zorluklar

Agile organizasyonla birlikte hayatımıza yeni birçok rol girdi, senin rolün bunlardan sadece biri. Peki, yeni yapıdaki diğer rollerle etkileşimlerin nasıl? Yeni yapıda yaşadığın zorluklar neler?

Chapter Lead rolü, çok hassas bir çizgide olan, duygusal zekâ, gözlem, empati ve geribildirim alma-verme becerilerinin üst seviyede olmasını gerektiren, koçluk becerilerinizde liderlik yaptığınız, ancak, geleneksel "yönetici" rolünden de uzak bir rol. Chapter'ınızda olan çalışanların tüm potansiyel, performans ve yetkinlik yönetimi, gelişimi ve değerlendirmesinden, Chapter Lead olarak sorumlusunuz. Ancak, Chapter'ınızdaki her bir üye, temelde sizin sahiplendiğiniz ve yönettiğiniz iş sonuçları için değil, farklı Squad'larda Product Owner arkadaşlarla birlikte Squad'larının temel iş öncelikleri için çalışıyor. Siz de paralelde üyesi olduğunuz kendi Squad'ınızdaki görev ve sorumluluklarınız haricinde, tabiri caizse, görünmez bir ağla spesifik bir uzmanlığı temsil etme

noktasında birbirinden beslenen insanların oluşturduğu bir yapıyı yönetmekten ve geliştirmekten sorumlusunuz. Bu, yönetmesi hiç kolay olmayan bir denge. Dolayısıyla, Chapter Lead olarak bizlerin, Chapter üyelerine ayırdığımız zaman kadar Product Owner'lar ve Tribe Lead ile paylaşımda bulunmamız, Chapter'ın öncelikleri, yenilikler, bariyerler ve gelişim alanları ile ilgili bilgi paylaşımı yapmamız gerekiyor. Bu yüzden, diğer tüm rollerle yakın bir birliktelik içinde çalıştığımızı söyleyebilirim. İletişim ve iş birlikteliği olmazsa olmaz.

Uzmanlık alanımdaki gelecek stratejilerini belirleme noktasında zorlandığımı söyleyemem. Bu konuda gerek Vodafone Grubu ve Vodafone Türkiye üst düzey yöneticileri, gerekse Tribe Lead'imiz ile yaptığımız bilgi alışverişi, beni epey besliyor. Zor olan kısım, bu stratejilerin tüm Squad'lara aynı oranda aktarılıp hayata geçirilmesi ve benim de buna zaman yaratmam. Bunu sağlamak için, ekipteki arkadaşlar ile mutlak surette iletişimde oluyorum. Basit haliyle, bir kahve molası, bire bir görüşmeler veya Chapter olarak bir araya geldiğimizde bu paylaşımı yaptığımız zeminler. Devamı ise takip gerektiriyor. Tüm ekip üyeleri aynı noktada olduğumuzu hissedene kadar ilgili konuda paylaşımlarımızı sürdürmeye devam ediyoruz. Bu role geçtikten sonra galiba en çok bocaladığım ve pratikte uygulamakta zorlandığım konu, Chapter'daki arkadaşlardan doğrudan Squad'larının önceliğine girmeyecek, ancak Chapter seviyesinde ihtiyaç duyduğumuz ve zaman ayırmalarını gerektirecek çalışmalar beklememdi. En başlarda, zaman ayırmalarını beklediğim konuları önceden belirtmek yerine aynı hafta içinde talep ettiğim için Product Owner ve Scrum Master arkadaşların haklı tepkilerini almışlığım oldu. Sonraki dönemde ise, Chapter üyelerinin Squad öncelikleri haricinde zamanlarına ihtiyaç duyduğum konularla ilgili onların Sprint planlamaları öncesinde konuyla ilgili iletişime geçmeyi ve Squad'larının iş listelerine almalarını sağlamayı yaşayarak öğrendim.

Chapter Lead, Tribe içinde belli bir sorumluluk ve yetkinlik alanının lideri. Ben DevOps Chapter'ına liderlik ediyorum. Açıkçası, içinde yaşadığımız dünya, gelecekle ilgili sürekli yeniliği ve değişimi bize dikte ediyor. İçinde bulunduğumuz dijital dünyada bu teknolojik dönüşüm bizim için bir zorunluluk. Bu kapsamda, bizim sergilediğimiz liderlik, değişimi yönetme çerçevesinde gerçekleşiyor. Ekibimizin sinerjisi teknolojik olarak hep yeniyi ve daha iyiyi istediğinden, bu konuda kendi Chapter'ım içinde bu mekanizmayı yönetmek zor olmuyor. Benim açımdan bunun en büyük zorluğu, özellikle geleneksel bir IT operasyon ekibinden devşirme bir Chapter olduğumuzdan, işleri Squad ekipleri içinde istediğimiz kalitede takip edebilmek ve bunun devamlılığını Squad ekipleri içinde sürdürülebilir kılmak. Bu yoğun tempoda ve sürekli kuralları değişen oyunda iyileştirmeler için fırsat yakalamak zor. Bir noktada sorumluluk Chapter Lead ve Chapter'da, ama işi planlama ve yapma Squad'da olduğundan, kopmalar yaşanabiliyor. Haliyle, Squad'ların kendi iş listeleri kalabalık. Onlar iş değeri üretmeye odaklanmış durumdalar. Biz ve hatta diğer Chapter'lar da bu öncelikler içinde kendi konularıyla yer almaya çalışıyor. Bu, bizim yönetmeye çalıştığımız en büyük zorluklardan biri.

Yalnız, bu tercihe dayalı bir şey olmaktan çıkalı çok oldu. Bu, bir zorunluluk ve biz de buna zaman yaratmak zorundayız.

Chapter Lead, Tribe içinde ekip yöneticisi olarak konumlanıyor. Tribe Lead'e sadece ekip yönetimi değil, strateji anlamında da destek veren kişiler olarak görülüyor. En büyük zorluk, Squad tarafından sahiplenilmeyen ya da fark edilmeyen işlerin Chapter Lead'ler tarafından toparlanmak zorunda olması. Squad'ların ticari hedefleri dışında kalan diğer işlerden bahsedebilirim. Örnek olarak, tüm Squad'ların daha iyi dijital analitik takibi (satışa dönen anlık bildirimler takibi) yapabilmesi için bir yetkinlik ihtiyacının netleştirilmesi gerekiyor. Bunun için 6 farklı Squad ve Dijital Analitik ekiplerinin ihtiyaçlarının anlaşılması ve çalışmanın koordine edilmesi gerekiyor. Bu çalışmanın farkındalığı Chapter tarafından yaratılıyor. Bunun dışında, deneyimdeki bozukluklar, erişilebilirlik, performans düzeltmeleri gibi işler için aksiyonların planlanması gerekiyor. Bu bağlamda, ben içinde bulunduğum Squad adına olan katkıyı sağlıyorum, ancak, diğer Squad'ların aynı şekilde sahiplenmesi için ekstra efor harcıyorum. Sanırım, zaman geçtikçe, Chapter'ımızdaki yetkinlikler arttıkça ve kendi uzmanlık alanımıza ilişkin büyük resmi hep birlikte daha net görebilir hale geldikçe, buradaki zorlukları da aşacağız.

Diğer bir konu da ürün ve pazarlama Chapter'ı alanındaki karar verilecek stratejilerin zaten Squad'ların önceliklendirme yaptığı konular olması. Bu yüzden, bu alandaki Chapter Lead'lere hareket alanı çok kalmıyor. Product Owner'lar ve bazı alandaki Chapter Lead'ler arasındaki denge konusu, sanırım, halen yol almamız gereken bir alan. Deneyip yanılarak bir yol bulmaya çalışıyoruz.

Agile öncesi dönemde klasik bir yöneticinin şapkasında bulunan ürün-insan sorumluluğu bu rolle birlikte ikiye bölünmüş durumda. Burada Chapter Lead'in rolü "insan" tarafını, başka bir deyişle yeteneklerin gelişimini üstlenmekten geçiyor. Bu doğrultuda, Tribe Lead ile organizasyonel konularda yakın çalışıyoruz. Eski yapıdan farklı olarak, artık Chapter içinde çalıştığım ve liderlik ettiğim üyeler farklı farklı Squad'larda çalıştıkları için onların çalışma şekillerini bire bir gözlemleme ile ilgili zorluklar artmış durumda. Ama, Agile çalışma şeklinin en büyük amaçlarından biri takımın kendi içinde otonom olması olduğu için, bu konuda "insan" tarafını ön plana alarak davranmaya son derece gayret gösteriyorum. Benim onları kontrol ettiğim bir yapının aksine, ihtiyaç olduğunda hemen yardıma koşabildiğim bir görevim var. Bu konuda tüm Chapter üyelerinin gelişim ve eğitim planlaması ile Chapter toplantılarındaki paylaşımların önemli olduğunu düşünüyorum. Henüz bunların önceliği konusunda istediğimiz noktada değiliz, ama bunu geliştirmek temel amacım olarak devam ediyor. Burada karşılıklı paylaşımlarla birlikte ileriye gidebileceğimize inanıyorum. Özellikle içinde çalıştığımız Squad'ın hedefleri doğrultusunda koşarken, arada kendimize dönüp gelişim noktalarımız üzerinde de çalışmanın, bizi çok daha iyi noktalara taşıyacağına olan inancım tam. Burada koçluk faaliyetleri, ihtiyaçlarımızın ortaya çıkarılmasında temel yöntem olarak kullanılıyor.

Chapter Kurgusunun Geleceği

Chapter yeni bir yapı. Bu yeni yapıyı gelecekte sence neler bekliyor?

Şirketimiz Chapter Lead rolünün bulunmadığı yaklaşık bir senelik bir dönemi de tecrübe etti. Buradan çıkan öğrenimler ile organizasyona bu rol dahil edildi. Gerek kişilerin yetkinlik gelişimi, motivasyon, potansiyel ve performans yönetimine getirdiği odak, gerekse Squad'lar arası spesifik uzmanlık seviyelerini temsil etme yetkinliği anlamında gözle görülmeyen bir bağın kurulması, bilgi ve deneyim paylaşımı açısından önemli bir rol üstlenmesi nedeniyle Chapter yapısına büyük görev düştüğünü ve bu yapının güçlenerek devam edeceğini düşünüyorum.

Gelecekte Chapter yapısı belki aynen devam eder, ama Chapter sayısının azalarak devam edeceğine ve etmesi gerektiğine inanıyorum. İleride bir zamanda uzmanlıkların gelişerek T-Shaped yetkinliğinin güçleneceğini, bu doğrultuda da yazılım geliştirme, DevOps, test gibi teknik Chapter'ların birbiri içine girerek tek bir Chapter'a dönüşeceğini düşünüyorum.

Chapter Lead'leri ekipleri bir arada tutan en önemli kişiler olarak görüyorum. Fakat, ürün ve pazarlama alanındaki Chapter Lead'lerin sorumluluk alanları Squad'lar ile çakışıyor ve pek hareket alanı kalmıyor. Bu sebeple, burada bir revizyon olacağını düşünüyorum. Organizasyon da sürekli değişen ve hatalarından öğrenen bir yapı. Zaten Agile'ın doğasında bu var.

Vodafone Türkiye ailesinin çalışanları olarak, uzmanlık ve kariyer gelişimimiz konusunda çalışmalar yapılmasının olmazsa olmaz olduğunu düşünüyorum.

Buna liderlik etme misyonu olan bir rolün her zaman olacağını, hatta öneminin gitgide artacağını düşünüyorum. Buradaki şekillendirme görevi de Chapter Lead'ler olarak bizlere düşüyor. Bu rolü ne kadar etkin kılabilirsek, şirketimiz ve tüm Vodafone Türkiye ailesi olarak bize o kadar değer katacağına inanıyorum.

Diğer yandan, Agile yapıya baktığımızda, yeni organizasyonun işleyen ana çarkını otonom Squad'lar oluşturuyor. Otonomiyi desteklemek için organizasyonda Agile Koç'lar var, tabii. Ancak, bu, tek başına yeterli değil, bence. Gerçek otonomi için yetkin takımlara ihtiyaç var. Yetkin bir takım da ancak yetkin bireylerden oluşabilir. İşte Chapter'lar olarak biz bu yetkin bireylerin gelişimi noktasında devreye giriyoruz. Bu nedenle, otonom ve istenilen değeri üretebilen bir yapının gerçekleşebilmesi için Chapter kurgusuna ihtiyaç var.

Başarılı Bir Chapter Nedir?

Senin için başarılı bir Chapter olmak ne demek?

Başarılı bir Chapter, Chapter üyelerinin kendi aralarında etkin bir iletişim ağından beslendikleri, çalışanların kendi Squad'ları özelinde hem kendi hem diğer rollerde rahat bir biçimde performans gösterebilecekleri bilgi ve yetkinlik seviyesinde oldukları, Chapter Lead'in Chapter üyelerine zaman ayırarak onlara her seviyede yatırım yaptığı ve gelişmelerine katkıda bulunduğu bir yapıdır. Chapter'ların toplamı eksiksiz bir Tribe yapısını oluşturduğu için, Chapter'ların yetkinlik gelişimi Tribe'ın toplam performansına yansıyacaktır.

Başarılı bir Chapter, kendi sorumluluk alanındaki işlerini yürüten, Squad'ını bu noktada yönlendiren ve harekete geçiren, kendi içinde bilgi paylaşımı mekanizmalarını kurarak işleten ekiptir. Önümüzdeki dönemde tamamen otomatize bir uygulama geliştirme çarkı olan, ekip içinde çapraz sorumluluk ve bakış açılarının geliştiği ve değiştiği ekipler olursak, kendimizi başarılı sayarım.

Mükemmelin sessiz kalması zordur. Başarılı Chapter'dan kaliteli ve sorunsuz iş çıkar. Kaliteli iş, hem insanları motive eder, hem de Tribe'ın iş sonuçlarına etki eder. Bu doğrultuda, motive insanlardan oluşan ve iş sonuçları üretebilen bir Tribe'ınız varsa, Chapter'larınız işlerini başarıyla yapıyor demektir.

Bence başarılı bir Chapter, Squad bağımsız yürüyebilen, bilginin şeffaf bir şekilde paylaşılabildiği ve her zaman değişime açıklıkla organizasyonunu ileriye taşıyacak bilginin toplandığı bir örgütlenmedir. Adaptasyon yeteneği ile her türlü hedefe koşma konusunda kendini geliştirmeye çalışan bir Chapter'ın başarılı bir Chapter olabileceğini düşünüyorum. Üyelerinin birbirine bağlı olduğu, paylaşım konusunda her zaman istekli, gelişen trendleri her zaman yakından takip eden ve yönetiminde olan ürünler/süreçler/teknolojiler konusunda uzmanlaşma isteğini hiç kaybetmeyen bir Chapter'ı başarılı bir Chapter olarak adlandırabilirim.

Chapter Lead'lere Tavsiyeler

- Agile çalışma şeklini benimsemiş ve bu yapı içinde Chapter yapısı kurma kararı alan şirketlere önerim, bu yapıyı halihazırda kullanmakta olan şirketlerden mutlak suretle geribildirim almaları, yapıyı yerinde gözlemleme şansları varsa eğer bu şanslarını kullanmaları ve profesyonel Agile danışmanlık hizmeti veren bir firmadan kuruluş aşamasında danışmanlık almalarıdır. Geçiş süreçlerinde halihazırda çalışmakta olan iç veya dış destek Agile Koç'lardan destek almak, hızlı ve efektif bir seçim olabilir. Üst yönetim desteğinin de ciddi şekilde alınması gereken bu süreçte, başta Chapter Lead'ler olmak üzere Chapter yapısına dahil olacak tüm ekip üyelerini insan kaynaklarının da desteğini alarak eğitimler ile yeni yapıya odaklamalı ve hazırlamalılar.

- Chapter'ın ne olduğunu ve ne şekilde çalışacağını çalışanlara önceden anlatmadan bu yapıyı kurmayın. Bizim örneğimizde tanım ve kişilerin bunu anlaması ve sindirmesi kısmı yeterli kalitede olmadığından, hikâyemiz bolca gözyaşı da içerdi. Özellikle kendi Chapter'ımda ekibi bir arada tutmak, bu belirsizlikte beklentileri karşılamak çok zor oldu. Bir de organizasyonun parçalı şekilde Chapter yapısına geçmesi, bence önemli ölçüde dönüşümü yavaşlatan bir kurgu. Geçiyorsanız, tam geçin, tüm Chapter'ları belirleyip aynı zamanda kurulumlarını tetikleyin.

- Belirsizliklerden korkmayın. Gri noktaların üzerine gidin. Chapter Lead olmak, kendini motive etmekten geçiyor. Ekibi bir arada tutmak için lider özelliklerini ön planda tutmak gerekiyor.

- Çok iyi bir Chapter Lead seçmelisiniz. Çok iyi bir Chapter Lead, gözlem yapabilen ve ekibinin mutlu bir şekilde katma değerli iş üretmek için çalıştığından emin olandır. Karışık ve birbirinden farklı senaryolardaki problemleri çözebilen kişidir. Dinleme ve durum analizi yapma yeteneği olandır. Değişik fikirlere açık olan kişidir. Farklı rolleri deneyimlemiş, ekibindekilerin yaptıkları işlerde derin ekspertizi olan ve empati yeteneği gelişmiş kişidir.

- Chapter'lar, çalışana ve onun gelişimine verilen değeri gösteriyor. Bu yüzden, Chapter'ların, insan odağını vurgulayan Agile çalışma şekli için kritik bir nokta olduğunu düşünüyorum. Agile dönüşüm planlayan şirketler için yetenek gelişiminin desteklenmesi ve otonomi açısından özenle bu yapıların kurulmasını tavsiye ederim.


#cometoagile

Tribe Lead ve Çevik Liderlik Kavramı

Tribe Lead, küçük çaplı bir şirketin genel müdürü gibi tüm sorumluluğu üstlenerek hedeflerini gerçekleştirmeye odaklanan bir rol. Ancak, bu rol, klasik anlamda bildiğimiz yönetici tanımından biraz daha farklı. Geleneksel liderlik kavramında karar verme, stratejik planlama, yönetsel beceriler öne çıkıyordu. Şimdiki yapıda ise Tribe Lead, Squad'ların en uygun şartlarda çalışmasından, kendi kararlarını verip uygulayabilecekleri ortamı yaratmaktan sorumlu.

"Zaman zaman arkadaşlarla kendi rolüm için şu tanımlamayı yapıyorum; geniş bir ailenin en büyüğü, büyük babası gibiyim. Aile içinde de öyledir ya, onlarca şey olur biter, ama en son babanın haberi olur. Ya da eğer gerçekten dertli biri varsa, işte birisinin harçlığı bitmiştir ve paraya sıkışmıştır, babaya gelir ister, ya da işte komşunun çocuğu ile kavga ettiyse, gelip yine babadan yardım ister. Aslında tam olarak benim bu dönüşümdeki rolüm böyle. Çok uzun zamandır kimseye neyi yapacağını direkt söylemiyorum. Daha çok, tavsiye isteyene tavsiye veriyorum, para isteyene para bulmaya çalışıyorum, yardım isteyene yardım etmeye çalışıyorum. Tam anlamıyla hizmet eden liderlik tarafındayım, ya da koçluk tarafındayım daha çok, diye özetleyebilirim."

Tribe'ın tanımına baktığınızda, belirli bir amaç ve ülkü birliği içinde hareket eden bir topluluktan bahsediyoruz. İşte bunu sağlamak, Tribe Lead'in görevi. Tribe'ın ana varoluş amacını oluşturma ve elindeki tüm imkânları bu yöndeki değerli iş çıktılarını elde etmek adına yönlendirebilme sanatından bahsediyoruz.

Tribe Lead'in doğru cevapları bilmesine gerek yok, ama doğru sorularla Squad'ın yol almasını sağlayabiliyor. Normal bir yönetici "Bunu yaparsak, ne olacak?", "Bunun finansal katkısı ne?", "Bu işten ne kazanacağız ya da ne kaybedeceğiz?" sorularına odaklanır. Başka bir deyişle, kararları biraz matematiksel ve sanal bir düzlemde projeksiyonlar. Alışık olduğumuz, işin içinde bizzat bulunan, detaya girip takip eden, her şeye müdahale eden bir tarz. Şimdi ise bundan bilinçli olarak uğraşarak vazgeçiyoruz[11]. Buna karizmatik liderlikten işbirlikçi liderliğe geçiş diyoruz. Amacımız, Squad'ı dataya ulaşmaya teşvik etmek, müşterilerle daha sık etkileşime geçirmek, farkındalıklarını artırmak. Squad'lar da aldıkları her aksiyonun sorumluluğunu hissediyorlar. Sonuçlarını şeffaf bir şekilde ortaya koyup, gerektiğinde hatalarından ders çıkarıyorlar. Bu yaklaşım, hem sahiplenmeyi, hem de öğrenmeyi destekliyor. Yöneticiler, ilham veren, rol model alınan liderlere dönüşüyor. Squad'lar, ondan karar vermesini değil de güven ve cesaret vermesini bekliyor. Bu anlamda, Tribe Lead daha fazla hizmet edecek bir rol; sürekli hizmet alan, hizmet isteyen bir rol değil. Bu, liderler için olduğu kadar Squad'lar için de bir kafa yapısı değişikliği demek. Bu yolda her geçen gün gelişiyoruz. İş yapışımızdaki dönüşüm ile liderlik kavramındaki dönüşüm kol kola gidiyor, birbirini besliyor.

Yeni liderlik modelinin altında koçluk becerileri yatıyor. Bu becerileri geliştirmek için hem içerideki koçlarımızdan hem de dışarıdan destek alıyoruz. Agile Koç'larla yakın çalışıyoruz. Tribe Lead rolü, biraz da "yaşayıp deneyimledikçe" gelişiyor. Her gün farklı bir durum ile farklı bir konu ile karşılaşabiliyoruz. Bunları çözdükçe, düşündükçe, hem organizasyon daha verimli hale geliyor, hem de kendimizi geliştirebiliyoruz.

Ayrıca, Tribe Lead'ler olarak, sık sık bir araya gelip, kendi içimizde deneyimlerimizi paylaşıyoruz. Karşılaştığımız sorunlarla ilgili birbirimizden görüş alıyoruz. Tribe Lead'lerin, değişime açık olması, merak etmesi, sürekli öğrenmesi ve kültüre katkı yapması gerekiyor. Sonuçta, bir kerede yapılan köklü bir değişimden bahsetmiyoruz. Bir başlangıç noktası var, sonra da sürekli öğrenerek geliştiriyoruz. Nasıl Squad'ın içinde ilgili bireyler hep birlikte dönüşüm yolculuğundan geçiyorsa, aslında Tribe Lead'ler olarak biz de bunun benzeri şekilde kısmen kendi içimizde yardımlaşan bir grup olarak birlikte ilerliyoruz.

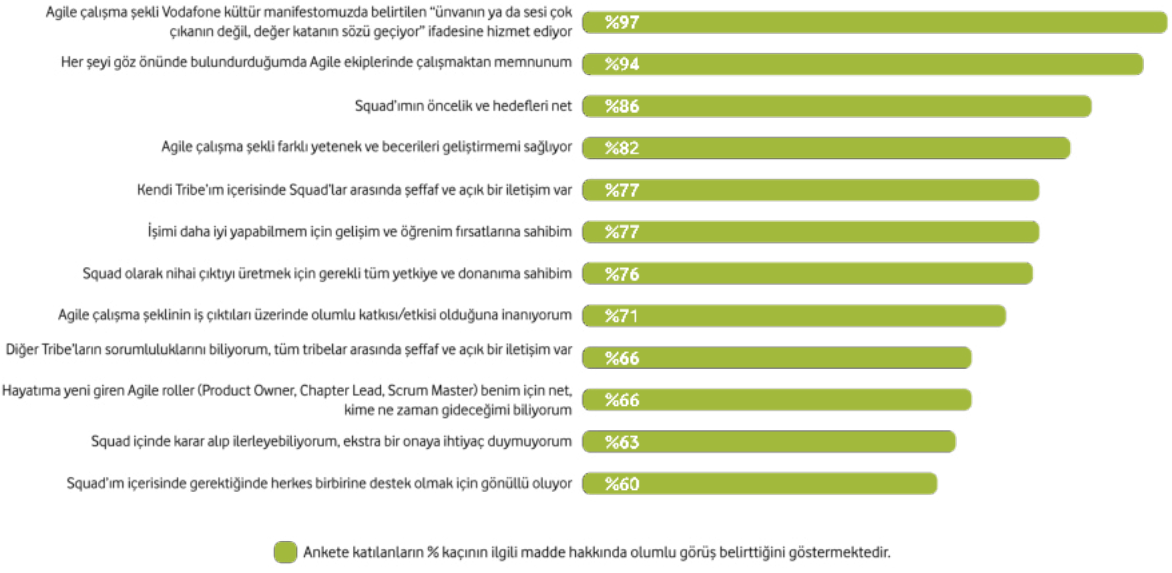
"Başlarda yaşadığımız bir deneyimden bahsedeyim. Bizim klasik iş yapış şeklimiz; ekip bir fikir oluşturur, birkaç hafta üzerine planları çalışır, ondan sonra gelir yöneticiye alternatifleri sunar, yönetici karar verir, şeklindedir. Verilen karara istinaden ekip 3-4 ay hazırlık yapıp yeni bir lansman yapar. Faturalı Hat Squad'ı bana geldi. Çok güzel bir strateji hazırlamışlar; 10'dan fazla alternatifi değerlendirmişler, 3 tanesini seçmişler, bir önerileri de var. Altı ay sonra hayata geçirmek istedikleri bu öneriyi onaylamam için konuyu bana anlattılar. Tribe Lead olarak ilk aklıma gelenler şöyleydi: "Ben aslında ikinci alternatifi beğendim, avantajları şu şekilde, çünkü diğerleri daha önce denendi. Zaten benim tecrübelerime göre...". Sonra bir adım geriye attım, derin bir nefes aldım, Agile kasları tekrardan çalıştırdım. Onlara ürün vizyonlarını, nereye gitmek istediklerini sordum. Bu vizyona giderken ilk adımınız ne olacak, bizi bu vizyona götüren ilk anlamlı çıktınız ne olacak, diye sordum. Daha sonra bir ya da birkaç Sprint sonra nerede olacaksınız, çıktıları tüketici ile nasıl tartıp evrileceksiniz, öğrenim planınız ne olacak şeklinde doğru şapkayı takıp sorgulamaya devam ettim. Ekip de şaşırdı. Belki de "O kadar direktör olmuşsun, biz onca öneri çalışmış birisini seçmişiz, bunu onaylayamayacak mısın?" diye düşündüler. O eski kaslarım hâlâ aktif aslında ve bu dönüşümde zor olan, o kafa yapısına dur demek. Karar vermek değil de ekibin kendi kararını verebilmesi için en doğru ortamı yaratmak."

[11]Tribe Lead 'in stratejiyi belirleyecek, sorunları hissedecek ve problemleri anlayacak kadar işin mutfağından gelen bir donanıma sahip olması elbette bir gereklilik. Ancak, bu bilgi ve deneyimin takımların işlerine karışır, onlara ne yapmaları gerektiğini söyler şekilde ön plana çıkmaması gerekiyor.

Güç Squad'da Artık!

Bu değişim sürecinin ve yeni yapının Squad'larda nasıl karşılık bulduğuna baktığımızda, şu çok açık ki Squad'lar mutlular. Bunu da yaptığımız anketlerden ve bire bir geribildirimlerden biliyoruz. Mutlular, çünkü bir amaç ve bunu uçtan uca yönetebilme alanı kazandılar. Herkesin ofise gelirken, belirli bir amacı var. Bu, ciro artırmak da olabilir, kullanıcı deneyimini iyileştirmek de olabilir, pazar payı alabilmek de olabilir. İşte bu net tanımlı amaç, insanları mutlu ediyor.

basit prensip ile çalışıyorlar. Bu Agile yapının aksine, çoğu klasik organizasyonda birisi "Stratejiyi ben yapacağım, sen masayı sil" der, bir başkası "Sildiğim zaman ben anlatayım, eksiklik olursa seni suçlayacağım" der. Agile yapı ile uçtan uca sahiplenme geldiği için, insanlar sorumluluğu daha çok hissetmeye başladılar ve mutlu oldular. Yeni yapı işin peşini bırakmamayı gerektiriyor. Çünkü, hiçbir karar tam doğru ya da yanlış olmuyor. Her zaman uygulamadan öğretiler çıkıyor. İlk yaptığınız Sprint üzerine ikinciyi yapmayı, üçüncüyü yapmayı ve işi



Bununla birlikte, sorumluluk alanları da çok net ve bu durum da insanları mutlu eden diğer bir unsur. Çok basit bir prensip aslında. Mesela, görevim masa silmek olsun. Bu örnek kapsamında, ben bezi de bulurum, silmek için stratejisini de oluştururum, masayı da silerim. Zaten, silmediğim yer de şeffaf bir şekilde görünür. Birisinin bana ne yapmak gerektiğini anlatmasına gerek yok. İşte Squad'lar bu çok

bırakmamayı getiriyor. Haliyle, sahiplenme duygusu da artıyor.

Bu çalışma şekli öğrenmeyi de çok artırdı. Bir karar verdin, bunu uyguladın ve sonuna kadar gittin, sonuç istediğin gibi olmadı ve başka yöne evrildin. Bundan daha iyi bir öğrenme şekli gerçekten yok. Başkasının aldığı kararı öğreneceğine, kendin aldığın kararın sonuçlarını

artısıyla eksisiyle görmek, çok daha öğretici. Ekipler, sürekli yeni şeyler denedikleri ve öğrendikleri, uzmanlıklarını genişleterek T-Shaped yönünde ilerledikleri için motiveler.

Tüm bunların yanında, hiyerarşinin azalması, önceliklerin net olması ve öncelikli işlere odaklanılması, başarının görülmesi ve paylaşılması, aynı amaca koşan kişilerin omuz omuza beraber çalışması, memnuniyet artıran diğer önemli faktörler oluyor.

Tabii, farklı rollerin bir arada çalışıyor olması, farklı beklentileri ve zorlukları da getirmiyor değil. Farklı disiplinlerden bir araya gelen arkadaşların özellikle başlarda uyum sorunları, fikir ayrılıkları olabiliyor. Öyle ki bu fiziksel çalışma ortamına bile yansıyabiliyor. Daha ticari taraftaki roller gürültülü çalışma ortamından çok rahatsız olmazken, yazılımcıların konsantrasyonları ciddi anlamda bozulabiliyor. Basit gibi görünen bu durum, sizin için önemli bir hal alabiliyor ve hep birlikte alınan karar neticesinde, bazı çalışma alanlarında sessiz çalışma saatleri uygulamasını başlatmanız gerektiği anlamına gelebiliyor.

Bir diğer önemli konu da sorumluluk alanları. Yeni yapının içinde birçok yeni rolümüz var. Bu rol ve sorumlulukların zamanla oturduğunu ve beklentilerin yeni yeni daha iyi karşılanır hale geldiğini söylememiz lazım. İlk başta, bu yeni roller gri alanlar yaratmıştı. Ancak, iş birlikteliğini öğrendikçe ve takım olarak hareket etmeyi başardıkça, bu gri noktalar zenginlik haline geliyor. Tabii, burada mümkün olduğunca, rollerin ana var oluş sebeplerini en baştan ortaya koyup iyi bir şekilde anlatabilmekte ve yolculuk sırasında da eğitim gibi faaliyetlerle desteklemekte fayda var.

İzolasyon sorunları ve bağımlılıklardan da kısaca bahsetmek gerekir. Ekiplere net bir amaç ve hedef doğrultusunda yön verip yetkilendiriyoruz ve koşmalarını istiyoruz. Ancak, bunu dedikten sonra, eğer ekip aşırı bir şekilde dışarıya bağımlıysa ve bu yüzden ilerlemede güçlük çekiyorsa, eskisinden daha kötü bir psikoloji yaşayabilir. Bu noktada, gücü devrettiğimiz takımların önünü açma görevi, yine Tribe Lead'ler olarak bizlere düşüyor.

Ya Squad Hata Yaparsa?

Burada hata yapmak ve riske etmek kavramlarına biraz değinmek gerekebilir[12]. Hata yapmak korkutucu olabiliyor, çünkü derinliği olan bir kelime. Bu derinliği iyi tartmak lazım. Amacımız, hukuki riskler almak, şirketin temel değerleri ve yapılarına aykırı şeylere kalkışmak değil. Öteki taraftan, risk boyutunu devamlı göz önünde tutup, günlük hayatımızda sanki her yaptığımız aksiyon şirketi batıracak gibi yaklaşmak, her şeyin bir hukuki ya da finansal boyunduruğu olduğunu düşünmek sınırlayıcı olur. Bu ayrımı yapmak gerekiyor. Tabii ki bizim de Tribe'larımız içinde hukuksal ve finansal sorumlularımız var. Sprint çıktılarımıza bakıyor, çıktıları yakından takip ediyor, belki ekibin sezemeyeceği riskli bir konu varsa proaktif olarak devreye giriyorlar. Ama bu, bir onay mekanizması değil, yakından takip ve proaktif uyarı mekanizması.

Agile yapının faydalarından bir tanesi, yeni organizasyon yapısı ile birlikte deneyselliğin resmi bir metodoloji olarak benimsenmesi oldu. Herkesin "Şurada bir etki var, ben bir bakayım buna" demesini teşvik etti. Çoğu öğretimizi biz böyle kazanıyoruz. Hep denir ya, hata yok öğrenmek var, diye, daha önce ne kadar desek de, bu çalışmıyordu. Mesela, bir iletişim yapıyoruz ve müşteri kabul ediyor, ancak, sonra vazgeçiyor ve yine eski tarifesine dönmek

[12]Hatasız Kul Olmaz!
Hatalarımızı açık yüreklilikle paylaştığımız, hep birlikte ders çıkarmak ve öğrenmek için bir araya geldiğimiz etkinliğimiz. Deneyselliği cesaretlendirmek için şirket içerisinde bir paylaşım platformu oluşturduk.

istiyor. Burada müşteri neden vazgeçiyor acaba? Normalde Agile yapıda olmasaydık, iş birimi olarak buradaki getiriye ve maliyete bakar, bırakırdık. Aynı örnek için teknik taraf olarak ise "Ben geliştirmeyi doğru yaptım mı, müşteriyi sürece doğru sokabildim mi, müşteri süreçten doğru çıktı mı" diye bakar ve yine bırakırdık. İşte bu ikisini birleştirdiğimizde ise, neden müşteri bu tekliften vazgeçti acaba, diye düşünmeye başlıyoruz. Bu soruyla birlikte durumumuzu irdelemeye başlıyor ve o zaman hatayı buluyoruz. Hatayı bulmak da o kadar kolay bir şey değil aslında. Yıllardır bunu böyle yapıyoruz dediğimiz noktada ezber bozalım diyoruz. Tabii, burada fırsat maliyeti var; öyle ki bazı bildiğimiz çalışan doğruları yapmaktan vazgeçiyoruz, alternatif fikirlere yatırım yapıyoruz. En katma değerli, en güzel hatalar da böyle bulunuyor. Bu açıdan, Agile bakış açısının getirdiği en güzel şey, emek ve zamanı farklı anlayışlar kazanabilmek için harcama güdüsü aslında.

"Deneme, yanılma ve öğrenmeye yönelik bir örnekten bahsedeyim. Dijital kanalımızda gençlere yönelik tarifeler yaptık. Genç müşterilerimiz tarifeleriyle birlikte istedikleri uygulamayı seçebiliyorlardı. Biz bu seçim özgürlüğünün satışları artıracağını düşünüyorduk. İstediğimiz gibi olmadı. Satın alma anında ekstra bir seçim aşaması koymak satın alma oranlarını düşürdü. Bunun yerine en çok tercih edileni sunmak ve satışı kapatmak, abonelerimizin seçimi satış sonrası özgürce yapmalarını sağlamak, bizi istediğimiz sonuca götürdü. Yeni ürün gamımızı bu yönde şekillendirdik. Bu öğretiyi diğer tarifelerimizde de uygulamaya başladık."

Gerçekten Otonomi Var mı?

Şunu söyleyerek başlayalım, hayat her zaman güllük gülistanlık değil. Otonomi, rol ve sorumlulukların temel çerçeve anlamında belirlenmesi ve el sıkışılması ile başlıyor. Evet, her Squad otonom olsun istiyoruz, ama özgürlükleri diğer takımların özgürlükleri ile sınırlı. Belli bir seviyede bağımlılık her zaman olacak. Squad'lar da minimum seviyede de olsa bu bağımlılıklarla yaşamayı ve iş birlikteliği ile bunları çözümlemeyi öğrenecek. Bununla birlikte, herkesin kendi alanı için yapması gereken şeyler var. Tribe Lead'in görevi bu bağımlılıkları ortadan kaldıracak kaynakları bulabilmek; bütçe gerekli ise bütçe bulmak. Mesela, bizim birçok açıdan Core IT'ye net bir bağımlılığımız var. Tribe'larımızdan birinde bunu azaltmak için gerekli bütçeleri yaratıp 6 kişilik minimum bir ekip kurarak Tribe içerisine bazı yetkinlikleri kazandırdık. Bu, bir kaynak bulma işiydi. Bu kaynağı bulup bağımlılıklarımızı azaltmak adına başka şeylerden vazgeçtik.

Bağımlılık yönetimi için Squad'ların ve Tribe Lead'in bakış açısından biraz bahsettik. Asıl önemli taraf, organizasyonun tasarımı. Bu tip bariyerlerin gittikçe minimize edildiği bir organizasyon tasarımı kurmak lazım. Bunun belki de sonu yok, sürekli evrilmesi gerekiyor. Mesela, bizim ilk bağımlılığımız iş zekâsı yetkinliği. Bu iş ilk kurulurken, herkes Tribe içinde iş zekâsı kadrosunun olmamasından şikâyet ediyordu. Eski yapıda bu yetkinlik fonksiyonel olarak finansa bağlıydı. Oturduk, finans ile birlikte bu işi çözdük ve bu yetkinliği ilgili Tribe'larımıza aldık. Pazarlamacı almayacağız, onun yerine bütçemizi iş zekâsı yetkinliğine harcayacağız, dedik ve hızla yeni arkadaşlarımızın işe alımını tamamladık. Bu arkadaşlar 2 ay gibi kısa bir süre içinde yetişti,

sisteme adapte oldular ve başarılı bir şekilde bu meseleyi çözmüş olduk. Tabii, bu mesele biter bitmez, yenisi geldi ve az önce bahsettiğimiz üzere, şimdi de Core IT ile birlikte orayı iyileştiriyoruz. Bu, hiç bitmeyen bir süreç. Ama, adım adım gerçekten çözülmesi gereken önemli bir konu.

Başarılı Squad'ın Özellikleri

Başarının tek bir tanımı yok[13]. Birden fazla unsur sizi başarılı yapabilir. Bir Squad'ın başarılı olması için her şeyden önce bir çıktı üretebilmesi gerekiyor. Bu yüzden, Squad'ın çıktı elde edebileceği bir kurguyu oluşturmak çok önemli. Squad'daki insan sayısı, Squad'ın yetkinlikleri, Squad'ın şirketin geri kalan kısımları ile olan bağımlılıklarının yönetimi gibi unsurları kastediyoruz. Bu unsurların doğru kurgulanması, gerektiğinde formasyonun değişen ihtiyaçlara göre tekrar şekillenebilmesi önemli. Squad'ın kendi ritmi ne ise o çerçevede çıktı üretmesi ve müşteriye dokunması gerekiyor. Sürekli teslimat yapabilen bir takım, başarıya giden yoldaki ilk adım.

Peki ya, hedeflerin tutturulması? Eğer hedefleri tutturamazsak, başarısız mı oluruz? Başarı tek başına hedefleri tutturmak değil. Tabii ki hedeflerimiz önemsiz demiyoruz, ama başarı tanımı doğru yolda gitmekle daha çok alakalı. Bu noktada, iki konu önemli. Birincisi, alınan yol. Mesela, diyelim ki bir Squad iş sonuçlarını iki birimden dört birime çıkarıyor, diğer bir Squad ise sekiz birimden dokuz birime çıkıyor. Burada sonucun dört ya da dokuz olması değil, ikiden dörde gitmiş olmak daha önemli. İkincisi ise doğru yolda ilerliyor muyuz? Hedeflerimiz birkaç ay tutmayabilir. Yeter ki doğru bir yol haritamız olsun, yeter ki adım adım küçük çıktılarla, üstüne koya koya devam edelim. Bu, takımın sahiplenmesiyle de alakalı. Squad olarak, var oluş amaçlarını sindirmişler mi, hedefleri sahiplenmişler mi, iyi bir takım olup farklı yolları deneyerek ilerlemeye çalışıyorlar mı? Yoksa bunların aksine, eski yapıda olduğu gibi işin kolayına kaçıp başka yerde suç bulmaya mı çalışıyorlar? Ben verdim, satış satamadı, fiyat yüksek oldu, pazar şöyle oldu, vs. bahanelerin arkasına mı gizlenmeye çalışıyoruz? Bizim yeni çalışma kültürümüzde deneyselliği ve farklı, yaratıcı düşünceyi tetiklememiz gerekiyor. Takım olarak hareket etmeyi beceren, kendi içinde birbirlerinin eksiklerini gideren Squad'larda bu kültürel bakış açısıyla birlikte çok iyi sonuçlar elde edilebiliyor[14].

Başarıyla ilişkili diğer önemli bir konu da yetkinliklerin gelişimi ve uzmanlıklar. Bunu da topluluklar ve Chapter'larla destekliyoruz. Her Squad çıktı üretebilir, ama o Squad'ın çıktı hızı, çıktısının kalitesi, ilk pazara çıkıştan sonraki adaptasyonu için barındırdığı insan kabiliyeti çok önemli. Artık, pazarlama ekipleri ile IT ekipleri bir arada. Squad'ın kaliteli iş çıkarabilmesi için o işle ilgili gerekli teknik kabiliyete sahip olması gerekiyor. Kullanıcı deneyimi, yazılım, performans, pazarlama gibi farklı uzmanlık alanlarında sahip olduğunuz insan kabiliyeti ve teknik beceriler, çok belirleyici.

Son olarak, Squad'lar arası harmoni ön plana çıkıyor. Vodafone'da 50'den fazla Squad var. Her biri müşteri deneyiminin farklı kısımlarından sorumlu. Bu Squad'lar ahenk içinde çalışınca, toplam müşteri deneyimi gelişiyor ve şirket olarak başarılı oluyorsunuz.

[13]Takımları birbirleri ile karşılaştırmaktan ziyade kendi içlerinde daha üretken çalışmaları için iyileştirme aksiyonlarına odaklanmak gerekiyor. En özet haliyle başarılı Squad; var oluş nedenlerini iyi anlamış, amaçlarına odaklanmış ve amaçlarının önündeki engelleri aşarak gelişen güçlü bir takımdır.

[14]Takım olarak hareket edebilme noktasındaki önemli bir faktör Squad'ın büyüklüğü. On kişinin üzerine çıkan rakamlarda, Squad'ın içerisindeki iletişimin negatif etkilenmeye başladığını deneyimliyoruz.

"Squad'ların Sprint bazında kendi hedeflerini belirledikleri bir yapı ve o hedeflerin ne ölçüde hayata geçtiğini çok rahat takip edebileceğiniz şeffaf bir kurgu olduğu için performansı gözlemlemek işin kolay kısmı. Performans konusunda iyileşmesini istediğiniz bir ekibe müdahale ise daha dikkatli olmak gereken kısım. Kendi adıma ilk baktığım, Agile çalışma yapısını ekip doğru anlamış mı, doğru uyguluyor mu, oluyor. Özellikle operasyonel bir kurgu içerisinden dönüşerek gelen ekipler, bu ritüel yapısını ve kendi katma değerlerini o ritüelleri tam uygulamadan anlayamayabiliyor. Sonrasında ise ekibe direkt müdahale yerine, tam aksine ekibi güçlendirmeye, onları hedefe nasıl ulaşabilecekleri konusunda cesaretlendirmeye ve hata yapmaktan çekinmemelerini sağlamaya çalışıyorum. Bunlardan sonra hâlâ bir gelişim ihtiyacı varsa, o zaman da ekibin yetkinlik anlamında desteklenmesi gerekir mi diye bakmak gerekiyor.

Çevik Liderlere Tavsiyeler

- Çevik bir liderin sabırlı, tutarlı, heyecanlı, istekli ve cesaretlendirici olması gerekiyor. Takımlara oyun alanı bırakmak ile hedefleri arasındaki dengeyi iyi kurmalı. Deneyselliği uygulatmalı, hataları öğrenmeye dönüştürmeli. Takımlarla bol bol vakit geçirmeli ve daima ulaşılabilir olmalı.

- Stratejinize ve yol haritanıza güvenmek istiyorsanız, hedeflerinizi takımlarınızla hep birlikte oluşturun.

- Liderlerin dirençli olabilme kültürünü oluşturmaları çok önemli. Aslında, herhangi bir anda organizasyona baktığınızda seksen tane iyi giden, yirmi tane de iyileştirilmesi gereken nokta çıkarabilirsiniz. Ne seksene bakıp sevinmek, ne de yirmiye bakıp üzülmek lazım. Bu, bir evrim işi, değişim sürekli devam edecek. Değişen şartlar içinde dirençli olabilmeyi başarmaktan bahsediyoruz. Yıkıldığında ayağa kalkmak, bir hedef tutmadığında öbürünün onun bir açığını kapatabilmesi ve yola devam edebilme kararlılığını ve dayanıklılığını göstermek önemli.

- Çevik lider, gezen liderdir. Koltuğunu bırak ve gezmeye başla! Eski yönetim tarzında belki de pek yapılmayan bir şey bu. Açık olmak, dinlemek ve mümkün olduğunca çok insana dokunabilmek gerek. Ekosisteminiz sayıca büyük olunca, her Squad'ın Sprint Review'larına ya da Chapter'ların toplantılarına girebilmek pek mümkün olamayabiliyor. Dolayısıyla, her saniyeyi insanlarla iletişim ve etkileşim kurmak, bağ yaratmak için fırsata çevirmek lazım. Önemli olan, olası en iyi şekilde her saniyede bir bağ kurabilmek, her saniye değerli bilgiyi almak ya da vermek. Bu da sürekli iletişim halinde olmaktan geçiyor. Biraz belki de arı gibi bütün insanları dolaşmanız, sürekli başka yerlerde oturarak gezmeniz gerekiyor.

- Tribe Lead, "ne" ve "nasıl" sorularından ziyade "neden" sorusuna cevabı olan bir lider olmalı. Tribe üyelerinin yetkinlikleri kadar başarılı, verimliliği kadar üretken olabileceğinin bilincinde, sorunları ortadan kaldırmak üzere çalışan, "hizmetkâr" bir lider olmalı.

- Çevik bir lider, ekiplerin hangi konularda desteğe ihtiyacı var, anlayacak kadar yakın, kendi özerkliklerini bozmayacak kadar da mesafeli olmalı. Takımlara dokunmak için en iyi pratik, Sprint Review'lara katılım

göstererek geribildirimde bulunmak.

- Agile dönüşümde amacın ritüellerden öte bir girişimci bakış açısı olduğunu unutmamak gerekiyor. Kurumların da hata yapma, belli bir fikre takılı kalmama, çalışanları yetkilendirme anlamında kendilerini dönüştürmeleri gerekiyor. Liderler için en büyük değişimlerden birinin, bir yandan iş sonuçlarından sorumlu iken, bu sonuçların nasıl hayata geçeceği kısmını ekibe bırakabilmek ve onların da fikirlerini ortaya koyup karar alabilecekleri bir ortam hazırlamak olduğunu düşünüyoruz. Squad'ların ve bu çatı altında çalışan herkesin girişimci bakış açısını kazanmalarını, yeni rollerini anlamalarını ve bulundukları Squad'ın ortak hedefini sahiplenmelerini sağlayabilmek için koçluk becerilerinin geliştirilmesi çok değerli.



Agile Koç Kimdir?

Agile Koç Kimdir?

Agile Koç; organizasyonların, takımların ve bireylerin değişime ayak uydurabilmeleri ve bu değişimi yönetebilmeleri için çalışır. Bu amaçla da çevik kültürün organizasyonun her seviyesinde içselleştirilmesi yolunda tüm kuruma koçluk yapar.

Haftaya göre değişim gösterse de, Vodafone Agile Koç'ları olarak;

- Koçluk verdiğimiz takımların aktivitelerine katılım,
- İhtiyaca göre bire bir görüşmeler,
- Agile kapsamında eğitimlerin verilmesi,
- Atölye çalışmaları (Bitti Kavramı nedir, MVP nedir, Ürün İş Listesi yönetimi, vb.),
- Şirket içinde iyileştirme topluluklarının kurulması ve işletilmesi (Scrum Master topluluğu gibi),
- Organizasyon seviyesinde koçluk (metriklerin oluşturulması, kültürel dönüşüme yönelik destekleyici adımların tasarlanması ve uygulanması, vb.)

yapıyoruz.

Vodafone'da Agile Koç'lar takımımızın adı "Agilizers". Agile pratiklerini organizasyon seviyesinde daha çok çalışana ve departmana ulaştırabilmek için 2019 başında İç Agile Koç Gelişim Programı başlatıldı. Vodafone tarihinde ilk kez, pozisyona özel olarak tasarlanmış değerlendirme ve seçme sürecinden geçen adaylar ile Agile Koç takımı oluşturuldu.

Bu seçme süreci 5 adımdan oluşuyordu:

- Niyet Mektubu
- Birinci Mülakat
- Bilgi Yeterlik Sınavı
- Değerlendirme Merkezi Uygulaması
- Son Mülakat

Bu süreçte, bu yeni role gönüllü başvurmuş adaylar, Agile bilgi seviyeleri, pozisyona olan istekleri, iletişim becerileri, koçluk yetkinliği, merak ve kendini geliştirme odağı gibi alanlarda değerlendirildiler.

Seçme sürecinden hemen sonra yaklaşık altı ay olarak tasarlanan Agile Koç Gelişim Programı başladı. Bu program kapsamında aşağıdaki eğitim ve sertifikalar bulunuyordu:

- Koçluk ve Geribildirim Eğitimi,
- Fasilitasyon Eğitimi,
- Çatışma Yönetimi Eğitimi,
- Scrum.org Professional Scrum Master Eğitimi ve PSM I Sertifikası,
- Scrum.org Professional Scrum Product Owner Eğitimi ve PSPO I Sertifikası,
- Hikâye Anlatıcılığı Eğitimi.

Bu program, gölge koçluğu ve birlikte koçluk aktivitelerini içeren iş başında öğrenme ile devam etti. Bu süreçte, programda yer alan yeni koçların, aktif olarak koçluk yapan arkadaşlarla birlikte takımların ritüellerine dahil olarak tecrübelerini artırmaları hedeflendi. Bu süreç, aday koçların aktif olarak koçluk vermesiyle devam ediyor.

Bir Squad'ın Koçluk Serüveni

Vodafone'da Spotify modeli uygulanıyor. Bu modelde, Tribe'ların içinde Squad'lar konumlandırılmış durumda. Her Tribe'ın şirket hedefine hizmet eden kendi hedefleri bulunuyor. Tribe'ların içinde bulunan Squad'lar da Tribe hedefine hizmet edecek hedefleri (OKR: Objective Key Results) düzenli olarak takip ediyor.

Yeni bir Squad'ın kurulum sürecini ele alırsak, öncelikle Squad kurulumlarında o takıma neden ihtiyaç duyulduğu, bu takımdan nelerin beklendiği, ne gibi değerler üreterek organizasyona katkıda bulunacağı belirleniyor.

Squad kurulumu dinamik bir süreç. Birçok Squad, Tribe ile birlikte en baştan kurulmakla birlikte, Tribe yaşamına devam ederken, ihtiyaç dahilinde de yeni Squad'lar kurulabiliyor ya da mevcut Squad'ların yapısında değişikliğe gidilebiliyor. Her koşulda, bir Squad'ın kuruluşunun ilk adımı olarak, bütün takım üyelerine mutlaka iki günlük temel Agile eğitimi veriyoruz. Bu eğitim sonrasında ise takımın ilk Sprint'ini koşmaya başlaması için takımla birlikte tüm gün sürecek bir etkinlikte bir araya geliyoruz. Bu etkinliğin ilk yarısında takım kurulumuna odaklanarak, takım üyelerinin birbirlerini daha yakından tanımaları için kaynaştırma aktiviteleri düzenliyoruz. Bununla birlikte, takımın ilk aşamada daha uyumlu ve aksamadan çalışmasını sağlayacak; takım adı, takım logosu, Sprint süresi, ritüellerin zamanları, Scrum Master'ın seçilmesi, takımın çalışma kuralları ve değerleri gibi temellerin atılmasını sağlıyoruz. Günün kalan kısmında ise ilk Sprint'in planlamasını yapıyor ve Sprint'i başlatıyoruz. Squad'ın Sprint koşmaya başlaması ile birlikte mutlaka takım olma ve geribildirim eğitimi düzenlenmesini sağlıyor ve kendilerine

aktif destek veren Agile Koç'lar olarak, bu eğitime takımla birlikte biz de katılıyoruz.

Ortalama bir Squad'ın olgunluğa erişebilmesi ve takım olarak temel Agile pratikleri aktif kullanır hale gelmesi için altı ay gibi bir süreye ihtiyaç duyuluyor. Bu süre zarfında Agile Koç olarak takımla birlikte yakın çalışıyoruz ve takıma hem Agile pratikler konusunda, hem de karşılaştıkları zorluklar noktasında destek vermeye devam ediyoruz. Ayrıca, bu süreç içinde üçer aylık periyotlarla Agility Check-Up (daha çok Scrum altyapısına adaptasyonu incelediğimiz ve iyileşme noktalarını bulmaya çalıştığımız bir değerlendirme), Team Health Check-Up (Squad'ın takım olma sürecindeki durumunu ortaya çıkarmak için yaptığımız bir değerlendirme) ve farklı birkaç yöntem ile Squad'ın ne durumda olduğunu takımın kendisinin fark etmesini sağlamaya çalışıyoruz. Aslında, şeffaflığı artırarak, takımın iyileştirme aksiyonlarını kendi alabilir ve ilerletebilir hale gelmesini tetiklemeye çalışıyoruz. İlk Sprint'lerde yaşanan zorlukların takımın birbirine alışması ve Scrum'ın da özünde olan ampirik süreçle beraber deneyerek ve yanılarak aşılmasıyla birlikte takımlara verdiğimiz desteği azaltıyoruz. Baktığımızda, yeni bir Squad'ın koçluk serüveninde takımın kurulumu, takımın ve takımdaki rollerin yeni çalışma şekline alışması, takımın birlikte hareket edebilmeye başlaması ve iyileştirme bakış açısının tetiklenmesi, ana odak noktalarımız. Takım olmak, kolay bir olgu değil. Çoğunlukla iyi bir takımınızın olması, altı aydan daha fazla zaman ve çalışma isteyen bir süreç. Ancak, şirket içinde yeni takım kurulumlarımız hızla devam ettiği için, bir noktadan sonra, takıma olan desteğimizi azaltmamız ve koçlar olarak yeni takımlara yönelmemiz gerekiyor. İşte bu noktada, bir Squad'a olan desteğimizi azaltmadan önce, en azından ilgili Squad'da takım ruhunun oluşumu adına pozitif kıvılcımları görmeyi ve takımın karşılaştığı problemleri kendi içinde çözebilecek olgunluğa gelmesini hedefliyoruz. Agile, ampirik süreç üzerine temellenmiş bir yaklaşım. Bu açıdan, takımın problemsiz çalışmasının değil, karşılaştığı problemlerle her defasında başa çıkmasının önemli bir gösterge olduğunu düşünüyoruz. Başka bir deyişle, KAIZEN kültürünün takım içinde ne kadar yer edindiğine ve içselleştirilebildiğine göre, takıma olan desteği azaltmaya başlıyoruz. Biraz açacak olursak, bunu değerlendirirken süreç içinde,

- Takım içi ilişkilerin durumunu,
- Takım içinde Bitti Kavramı, vb. standartların aynı seviyede sahiplenildiğini,
- Scrum Takımı için gerekli rol ve yetkinlik güçlendirme programlarının sağlandığını,
- Squad/Tribe hedeflerinin Squad içinde anlaşıldığını,
- Karşılaşılan engelleri takip edebilmek ve çözmek için neler yapıldığını,

gözlemlemeye çalışıyoruz. Bu gözlemleri, belirli periyotlarla yaptığımız check-up'lar ile destekleyip takımın belirlediği önceliklere göre, gerekli aksiyonların alınmasını sağlamaya çalışıyoruz.

Bir takım yukarıda belirttiğimiz konuları içselleştirerek aksiyon alıp takip ediyorsa, bizim için "olgun" bir takımdır.

Her Squad "Takım" Olabilir mi?

Bir Squad'ın Agile anlamında takım olduğunu söyleyebilmemiz için aşağıdaki başlıca özelliklere sahip olmasını bekliyoruz:

- Tüm takım üyeleri tarafından çok iyi şekilde anlaşılmış olan hedefler ve vizyon,
- Takımdaki tüm üyelerin takım hedefleri doğrultusunda tutkuyla çalışarak sorumluluk alması ve yardımlaşması,
- Takım üyelerinin birbirine gelişim alanları ile ilgili olarak açık geribildirim verebilmesi,
- Takım üyeleri arasındaki güven ve saygı ortamı,
- Sürekli olarak gelişimi hedefleyerek mükemmelliğe ulaşma çabası,
- Hep birlikte takım önündeki engelleri kaldırmak ve kendi kararlarını alabilmek adına cesaretle çalışmak,
- Takımda müşteriyi odağa koyan bir anlayış geliştirilmiş olması.

Takım olmak, zorlu bir süreç. Yüksek seviyede iletişim, şeffaflık ve takım üyeleri arasında iş birliği gerektiriyor. Bazı kişiler takım halinde çalışmaya yatkın olabilirken, diğer kişiler bunu pek tercih etmeyebiliyorlar. Farklı iş yapış alışkanlıkları, farklı karakter özellikleri ve farklı beklentiler dikkate alındığında, Agile dönüşüm ile birlikte farklı disiplinlerden gelen ekip üyeleri arasında anlaşmazlıklar ortaya çıkabiliyor. Bu, aslında doğal bir patinaj. Öyle ki bugüne kadar bireyselliğin daha ön planda olduğu bir çalışma şeklinden bir anda takım olmaya geçmek, kolay bir adım değil. Ancak, takım olma yolundaki patinajlar her zaman üretilen değere etki etmese de, takımın ahengini bozup uzun vadede daha büyük sorunlara yol açabilir. Bunu engellemek için, takım oluşturulurken, Agile değerlere vurgu yapılarak takım açık iletişime yönlendirilebilir, farklılıkların doğallığı kabul edilerek hedefler doğrultusunda ilerlenmesi kolaylaştırılabilir.

Ayrıca, ekip üyeleri birlikte yan yana çalıştıkça ve birlikte sonuçlar üretmeye başladıkça, bu problemler doğal seyrinde azalabilir. Üstesinden gelinen zorluklar ve küçük de olsa edinilen başarılar, takımın kaynaşması için önemli ilerlemeler. Bunlarla birlikte, koçlar olarak, takım olmayı ve bu doğal kaynaşma sürecini hızlandırmak amacıyla, iş dışında da takımın bir araya gelmesini sağlayan aktivitelerle bu süreci desteklemeye çalışıyoruz.

Tabii, bu her zaman çalışmayabiliyor. Takım içinde anlaşmazlık yaşanmasına ilişkin yaşadığımız ve çözemediğimiz bir örnekten bahsedelim. Öncelikle şunu belirtmekte fayda var, bahsedeceğimiz takımın oluşum aşamasında Agile değerleri tartışmaya yeterli vakit ayıramadık. Belki de bu, en baştan yaptığımız önemli bir hataydı. En baştan, takım olma zeminini konuşmayıp, arkasından takım içinde anlaşmazlıklar yaşanmaya başlayınca, farklı yaklaşımlar uygulamak durumunda kaldık. Takım içi patinajı fark ettiğimizde, ilk aksiyonumuz, akla gelen en basit yöntem oldu; anlaşmazlık yaşayan takım üyelerinin bu sorunlarını kendilerinin çözmesi için kendi içlerinde bire bir görüşmeler yapmalarını sağlamak. Ancak, bu koçluk yaptığımız görüşmelerden istediğimiz sonucu alamadık. İkili tartışmaların yaşanmaya devam etmesi nedeniyle, takım içi dinamikleri tetiklemek için "Takım Olma ve Geribildirim" eğitimi düzenledik. Bu eğitimde takım çalışması ve geribildirimin ne kadar değerli olduğunu oyunlar ve örnekler ile ele almaya çalıştık. Fakat bu da bir sonuç üretmedi. Sorunların hâlâ devam etmesi üzerine, bunun tüm takımı etkilediğini de göz önünde bulundurarak, ilgili Chapter Lead'lerle sorunu paylaşma yolunu seçtik. Aslında koçlar olarak biz de yardım istedik. Tüm bu eğitimler, bire bir görüşmeler ve çalışmalar neticesinde sorunun aslında işlerinde gerçekten

çok başarılı iki takım üyesinin birbirlerine zıt karakterleri nedeni ile anlaşamaması olduğu ortaya çıktı. Futboldan örnek verecek olursak, son yılların en iyi takımlarından biri olan Barcelona'ya yine son yılların en "değişik" santraforu olan Zlatan İbrahimoviç'in katılması haberi, Barcelona'yı daha da durdurulamaz hale getirecek diye düşünmüştü herkes. Fakat daha ilk sezonunda, İbrahimoviç'in üstün yeteneklerine rağmen Barcelona'ya uygun olmadığı anlaşıldı ve sezon sonu İtalya'nın yolunu tuttu. Bizim yaşadığımız örnek de buna çok benzer bir senaryo. Son durumda yapılacak şey aslında kimsenin daha fazla mutsuz olmasına izin vermeden zoraki birlikteliği sonlandırmak olacak gibi görünüyor.

Bu örnekte de gördüğümüz üzere takım olmak, takım ruhunu yaratmak gerçekten zor. Ne yaparsanız yapın, ne kadar çabalarsanız çabalayın, bazen de mayanın tutmayacağını kabul etmek gerekiyor.

Agile Olduk mu?

Buna "evet" demek pek mümkün değil. Agile yaklaşımların temel felsefesi deneysellik olduğu için, şirket içinde sürekli yeni yöntemleri tecrübe etmeye gayret gösteriyoruz. Bu yöntemlerde zaman zaman başarısız da oluyoruz, ancak, önemli olan, hata yapmaktan korkmayacak bir ortamın gelişmesi ve bunun norm haline gelmesi. Bu yüzden, belki de "Olduk" demek, zaten bu kültürün doğasına aykırı. Olduğunuzu düşündüğünüzde, yeni bir şeyler deneyerek ilerlemeniz gerekiyor. Bu doğrultuda, kültürel anlamda henüz önümüzde uzun bir yol var diyebiliriz.

Agile yolculuğumuza baktığımızda, şu anda odaklandığımız iyileştirme alanlarını;

- Kültür ve bireylerin dönüşümü,
- Değerlere odaklanmadan, mekanik Agile uygulamalarının önüne geçmek,
- Bağımlılıkların azaltılması,

olarak belirtebiliriz.

Takımların temel yapıtaşının bireyler olduğunu da göz önünde bulundurursak, kişilerin çevikliği de bizim ana odak noktalarımızdan. Bunun için, ihtiyacı gördüğümüzde, mümkün olduğunca, bire bir koçluk yapmaya çalışıyoruz. Burada önemli olan nokta, kişilerin değişime ne kadar açık olduğu. Bireyler değişmeye başladığında takımlar, takımlar değişmeye başladığında organizasyon değişmeye başlıyor. Agile dönüşümü değerli kılan ise bu değişimin, organizasyonun olağan dinamiği haline gelerek sürekli olması. Genel anlamda, Agile değerler, şeffaflık ve müşteri odaklılık ekseninde bir çalışma ortamı oluşturulmasına destek oluyoruz. Bunlar sağlandığı sürece, neyi nasıl yaptığımız, aslında geri planda kalan şeyler. Bu değerleri ve prensipleri gerekirse sürekli tekrarlamak ve hatırlatmak, yeni pratiklerin keşfedilmesini sağlayacak ortamı oluşturmak, işin temeli. Bu noktada, Chapter Lead'ler ve Tribe Lead'ler önemli paydaşlarımız. Bu roller ile birlikte çalışarak, önce onların Agile değerlere sahip çıkan hizmetkâr liderlik özelliklerinin gelişimini, dolayısıyla da onların temas halinde olduğu bireylere örnek teşkil etmelerini sağlamaya çalışıyoruz.

Şunu açıkça söylemek gerekiyor ki uzun yıllar eski sistem ve kültür içerisinde yoğrulmuş yöneticilerin bu yeni çalışma şekline uyum gösterip göstermediği başarıyı yakalamak açısından çok önemli. Eğer bu dönüşüm sağlanamazsa yöneticilerin talepkar davranışları sistem içerisindeki birçok rol üzerinde devam ederek zorluk yaratır. Biz de bu tarz zorlukları yaşadık. Yılların öğreti ve alışkanlıklarını bir anda değiştirmek kolay değil, elbette. Bu dönüşüm sürecini desteklemek adına eğitimler, çalıştaylar ve bire bir koçluk çalışmaları ile ilerliyoruz.

Ayrıca, şirket genelindeki engellerin aşılması noktasında Scrum Master'lar da bizim için önemli, gücümüze güç katan bir rol. Henüz organizasyonumuzda Scrum Master rolünün güçlü bir etki sağlayabileceği bir çalışma şeklini oluşturmak noktasında zorluk yaşıyoruz. Bunu aşmak ve bizimle birlikte hareket eden değişim elçilerinin sayısını artırmak için Scrum Master Topluluğu gibi oluşumlarla organizasyon seviyesindeki farkındalığı canlı tutmaya çalışıyoruz.

Bunların yanı sıra, işin kültür boyutundan daha mekanik seviyeye dönüp bakalım. Tüm Squad'larda Agile pratikler istendiği gibi çalışıyor mu? Elbette hayır. Hepimizin uzun yıllar geleneksel yöntemlerle çalıştığı bir düzenin ardından, büyük bir dönüşüm ile yeni kaslar geliştirmeye çalıştığı bir yolculuktayız. Haliyle bu yolculukta zaman zaman baş etmemiz gereken aksaklıklarla da karşılaşabiliyoruz. Bu aksaklıklardan ve direnç noktalarından belki de ilki, günlük aktivite olan Daily Scrum'a karşı oluşan direnç. Bunun altında takımın dinamiklerine bağlı olarak farklı nedenler olabiliyor. Örneğin:

- Zaten sürekli bir aradayız, her sabah toplanmaya gerek yok.
- Bugün, işlerin durumunda herhangi bir değişiklik olmadı, konuşmamıza gerek yok.
- Daha dün planlama yaptık, detaylı olarak işleri konuştuk. Bugün tekrar konuşmamız gerekmez.

Bu tip durumlarda öncelikle takıma bu ritüelin faydasını aktarıyoruz. Takımın yine de direnci devam ediyorsa, bir süre gözlemliyoruz. Ritüelin yapılmamasından kaynaklı bir aksaklık gözlemlediğimizde ise bunun tartışılması için bir platform oluşturup, takımla birlikte Daily Scrum'a olan ihtiyacı ve bu pratiğin faydalarını yeniden değerlendiriyoruz.

Bizler Agile Koç'lar olarak, öncelikle takımın yanında olup direncin öz sebebine odaklanıyoruz. Takımın, aktivitenin katacağı faydayı anlamasını ve bunu deneyimleyerek sahiplenmesini sağlamaya çalışıyoruz. Bunu takıma doğru sorular sorarak ve uygun takım oyunları ile pekiştiriyoruz.

Agile dönüşümün önündeki en büyük engellerden biri de takımlar arasındaki bağımlılıklar. Vodafone gibi büyük organizasyonlarda geleneksel yapıdan Agile yapıya geçerken, bu bağımlılıklardan tamamen kurtulmak mümkün olmuyor. Yeni bir takım kurarken mümkün olduğunca gerekli tüm rolleri takıma getirmeye çalışsak da, pratikte farklı nedenlerle bazı bağımlılıklar oluşabiliyor. Bağımlılıkları gidermek veya minimize etmek adına organizasyon olarak aksiyonlar alıyoruz. Örneğin, Tribe içinde birçok takımın bağımlılık geliştirdiği bir takımın üyeleri, bağımlılıkları yok etmek amacıyla Tribe'taki diğer takımlara dağılabiliyor. Tribe dışındaki bağımlılıklar belirli bir alana ya da takıma yoğunlaşıyorsa, bu bağımlılıkları minimize etmek amacıyla belirlenen takım Tribe içine taşınabiliyor. Teknolojide ise adım adım mikroservis mimarisine geçişi planlıyor ve gerçekleştiriyoruz.

Diğer bir engel ise dönüşüm yolculuğuna çıkmadan önce el sıkıştığımız, farklı lokasyonlarda geleneksel yöntemlerle çalışarak hizmet veren kâr ortağına sahip takımlarımız var. Bu takımlarımız dönüşüm öncesi el sıkıştığı sözleşmelerinden dolayı kâr ortakları ile aynı lokasyonda ve aynı hedefe koşamıyordu. Product Owner'ların öncelikleri ile kâr ortaklarının öncelikleri farklılaşıyor, Product Owner'ın önceliğine saygı duyulmadığı için istenilen Sprint hedefi tehlikeye girebiliyor. Squad'ın aynı lokasyonda çalışmaya başlamasını sözleşme nedeniyle sağlayamasak da tüm takımın öncelikle aktivitelere uzaktan da olsa katılmalarını sağlayabileceğimizi gördük. Bunu kâr ortağımızın ve yönetimin bizi anlaması için yaşadığımız sorunları resmederek anlattık. Artık, planlamalardan önce Ürün İş Listesi İyileştirme ritüelleri yapılarak hem planlanacak maddelerin

detayları, hem de bunların öncelikleri konusunda ortak noktaya gelebiliyoruz. Takım üyeleri iki farklı şirket çalışanlarından oluşsa bile, ortak bir hedefe ilerlenebileceğini görüldükçe, daha fazla takım olmaya ve bu anlamda iyileşmeler sağlanmaya başladı. Önümüzdeki dönemde hizmet sağlayıcıların aynı lokasyonda olmasını dikkate aldık. Sözleşmelerin yenilenmesi ve imzalanması sırasında bu konuya dikkat edilmesini sağladık. İçeride geliştirilmesi mümkün olan yetkinlikler için gelişim faaliyetlerine başladık.

Roller Arası Etkileşimler

Vodafone Agile dönüşümünde organizasyonda büyük değişiklikler yapıldı. Daha az bürokrasi ve hiyerarşinin sağlanabilmesi için daha yatay bir organizasyona evrilmeye çalışılıyor. Squad'lar içinde dönüşümün sancıları belki organizasyon seviyesindeki kadar şiddetli değil, ama kendi kendine organize olan bir takım olmanın ve bunu şeffaflıktan ödün vermeden gerçekleştirmenin ne kadar zor olduğunu görüyoruz. Bu kapsamda, takımda rol ayrımı gözetmeden, tüm takım üyeleri ile mümkün olduğunca yakın çalışmaya ve Agile prensiplerini içselleştirmelerine yardımcı olmaya gayret ediyoruz. Agile'ın ilk değeri kişiler ve kişiler arasındaki etkileşimi işaret ettiği için, takım içi ilişkilere odaklanarak takımla birlikte gelişim alanları belirliyoruz.

Agile Koç'lar olarak, bu dönüşüm sürecinde Squad'ların yanı sıra Tribe bazında da sorumluluklarımız var. Yakın zamanda Agile Koç'lardan oluşan Agilizers Squad'ının da önceliklerinden biri, takım seviyesinde olan dönüşüm desteğinin daha geniş bir yelpazede yapılmasını ve sürdürülmesini sağlamak. Tribe içerisine baktığımızda, Scrum çerçevesinde yer alan roller dışında, organizasyonun ihtiyacına göre ortaya çıkan Chapter Lead ve Tribe Lead gibi roller de yer alıyor. Bu roller arasındaki etkileşimi artırabilmek adına Scrum Master-Tribe Lead, Tribe Lead-Product Owner-Chapter Lead görüşmeleri ve Tribe Day'ler gibi etkinliklerin düzenlenmesine katkı sağlıyoruz. Tribe Day'lar, Tribe'ların organizasyonda bulunan paydaşlarını davet ettikleri bir oluşum. Temel olarak, Tribe'da bulunan takımların yaptıkları iş çıktılarını ve paydaşlardan beklentilerini dile getiriyorlar. Tribe Day'in bitiminde ise "happy hour" aktivitesi ile sosyalleşecek güzel bir ortam oluşuyor.

Chapter Lead, organizasyon olarak anlamakta en zorlandığımız rol diyebiliriz. Bunun nedeni, pratikte uygulanışı ile organizasyonda tanımlandığı şeklinin bire bir örtüşmüyor olması. Chapter Lead'in birincil rol ve sorumluluğu, Chapter'da bulunan üyelerin yetkinlik seviyelerini artırmak. Bu doğrultuda, Chapter üyeleri ile düzenli bire bir ve ekip toplantıları yaparak yetkinlik seviyelerini anlayabilmesi ve bu seviyeleri yükseltmek için gerekli aksiyonları alabilmesi yönünde bir beklenti var. Organizasyonda ise daha çok İnsan Yöneticisi olduğu yönünde bir algı oluşmuş durumda ve bu algıyı değiştirmek için birtakım çalışmalar yapıyoruz. Chapter Lead'ler ile rol tanımına uygun şekilde çalışmalarını sağlamak için neler yapabileceğimizi değerlendiriyor ve onlara buna uygun bir platform oluşturmaya çalışıyoruz.

Aynı zamanda, diğer Tribe'lar ve Agile çalışmayan birimlerin iletişimini desteklemek üzere farkındalık seansları ve eğitimler düzenlenmesini sağlıyoruz.

Agile Tavsiyeler

Yeni kurulacak bir Squad için:
Agile yaklaşımlar, sizin ne yapacağınıza değil, zaten uzmanı olduğunuz işi nasıl daha etkin, daha rahat, daha verimli yapacağınıza ve takımca birlikte keşfetmenize yardımcı olur. Bununla birlikte takım sözleşmesi yapmak ve bu sözleşmeyi belirli periyotlarla tekrar gözden geçirmek faydalı olacaktır.

Takım sözleşmesi, takım üyeleri tarafından kabul edilen temel kurallardan oluşuyor. Bu kurallar, takımdaki herkes tarafından net olarak anlaşıldığında, basitlik ve efektif olma yönünde takıma büyük katkı sağlıyor. Takım sözleşmesi için, ritüellerde laptop ve telefon açmamak, takım Daily Scrum saatinin sözleşmeye yazılması ve her Sprint'te en az bir Retrospektif aksiyonunun hayata geçirilmesi gibi birkaç örnek sıralayabiliriz.

Yeni bir Scrum Master için:
Değişime, yeni fikirlere, yeni şeyler deneme cesaretini göstermeye, Scrum çerçevesinin doğru uygulanması için gerektiğinde yönetimin ya da takımın karşısında durmaya hazır olmalı. Bu süreçte çok okuyarak, araştırarak kendini geliştirmeli[15].

Yeni bir Product Owner için:
Takımın yöneticisi değil, bir parçası olduğu bilincine sahip olmalı ve bu doğrultuda hareket etmeli. Takımı değil, ürünü yönettiğinin bilincinde olarak, ürününü, pazarını, rakiplerini, müşterisini ve paydaşlarını çok iyi tanımalı. Bunları yaparken değere odaklanmalı ve bunu takıma da hissettirmeli.

Yeni bir Tribe Lead için:
Öncelikle kendi davranışlarını değiştirmeli ve Agile lidere dönüşmeli. Bu dönüşümle birlikte, takımların kendi kendilerini yönetebilecekleri ve yaratıcı olabilecekleri bir ortam oluşturmalarını desteklemeli. Takımdaki tüm üyeler için kapısını rahatlıkla çalabilecekleri ortamı yaratabilmeli. Tribe Lead'lerin, en başından itibaren, Product Owner, Chapter Lead ve Scrum Master'ların yanında, diğer tüm Squad üyeleri ile de irtibatta olması çok değerli. Takımların Sprint Review'lerine paydaş olarak katılarak, ortaya çıkan ürünle ilgili geribildirimlerini aktif olarak vermeli ve bu anlamda takımın gelişimine katkı sağlamalı.

Yeni bir Chapter Lead için:
Alanında uzman olduğu kadar, bu yetkinliğini ve bilgisini başkalarına da aktarabilen ve onların kendilerini geliştirmelerinin önünü açan kişi olmalı. Bunu yaparken Chapter üyeleri ile yakın temas içinde olarak onların gelişimine de bire bir katkı sağlamalı. Bu teması sağlamak üzere düzenli bire bir görüşmeler ve haftalık Chapter toplantıları organize etmeli. Chapter'da özellikle bireysel yetkinliklerin artırılmasına yönelik aksiyonları, maddeler halinde öncelikli bir liste şeklinde takip etmeli.

Agile dönüşüm yapacak yeni bir organizasyon için:
Dönüşüme en baştan CEO seviyesinde farkındalık ve sahiplenme ile başlayın.

Neden değişmek istediğinizi, hedeflerin ne olduğunu ve insanların bundan nasıl etkilenebileceklerini açık bir iletişimle ortaya koyun.

Çeviklik, sadece takımların bazı pratikleri hayata geçirmesiyle elde edebileceğiniz bir olgu değil. Dönüşüme başladığınızda, çalışmalarınızı tek seviyede (takım seviyesi) bırakmayın. Özellikle yöneticilerin dönüşüme dahil olmasını, onların davranışlarında bu değişimi başlatarak liderlik etmelerini sağlayarak ilerleyin.

Sürecin en başından iç Agile Koç'larınızı yetiştirmeye başlayın.

Organizasyon yapınızı kurarken ve rolleri belirlerken, mümkün olduğunca, şirket içindeki

[15]Kitap Önerileri:
The Agile Samurai: How Agile Masters Deliver Great Software (Jonathan Rasmusson)
Agile Project Management with Scrum (Ken Schwaber)

herkesin dahiliyetini sağlamaya çalışın. Tepeden inme bir tasarımı tutundurmak, kişilerin bunu içselleştirmesini sağlamak, daha zor bir yöntem.

Agile ile birlikte yeni gelen rollerin (Product Owner, Scrum Master, vb.) eğitimi ve yetkinliklerinin geliştirilmesi için en baştan destekleyici planlar yapın.

Herkes yapısal olarak Agile bir takım içerisinde çalışmayabilir ancak bu Agile prensiplerin yaşatılması için bir engel değil. Şirket kültürünü ilerletmek ve dili değiştirmek için adımlar atın.

Agile Koç olmak isteyenler için:
Agile felsefesini anlayarak, benimseyerek ve hayatlarının her noktasına yansıtmaya çalışarak başlayabilirler. Ne kadar uzun yıllar bu işi yapsalar da, bir taraflarının hep bilgiye aç bir öğrenci olarak kalması gerektiğinin de farkında olmaları gerekir.

BOL ŞANS DİLERİZ!


vodafone

Bilgisayar Bilimlerine Giriş- Ders İçeriği

- Bilgisayar Ölçü Birimleri- Temel Kavramlar
- Boole Cebri
- Algoritma ve Akış Şemaları
- Programlama Dilleri
- İşletim Sistemleri
- Bilgisayar Ağları ve İnternet Teknolojisi
- Sayı ve Kodlama Sistemleri
- Yazılım Mühendisliği
- Veri Yapıları ve Veri Modelleri
- Veritabanı Yönetimi ve SQL
- Yazılım Modelleme Dilleri

Haftanın Konusu

- Bilgisayar Ölçü Birimleri
- Temel Kavramlar-Alınacak Dersler
- Bilgisayar/Yazılım Mühendisliği Eğitimi ve Görevleri
- Bilgisayar/Yazılım Mühendisliği Konuları
- Bilgisayar Donanımı
  - Merkezi İşlem Birimi
  - Bellek
  - Giriş/Çıkış Birimleri
  - Bilgisayar Ağları
- Bilgisayar Yazılımı
  - Yazılım
  - İşletim Sistemi
  - Algoritma/Programlama Dilleri
  - Veri yapıları/ Veritabanları
  - Veri Güvenliği
- Özet/Sorular/Tartışma

Bu bölümde bilgisayar bilimi ve yazılım mühendisliği hakkında temel kavramlar verilmeye çalışılmıştır. Öncelikle bilgisayar ve yazılım mühendislerinin görevleri açıklanmakta, ardından temel konular ve tanımlar verilmektedir.

Bilgisayar Ölçü Birimleri-Nedir?

Bilgisayar ölçü birimleri, bilgisayarın hafızasında (RAM) işlenen, veri kayıt ünitelerine (Hard Disk, DVD vb.) kaydedilen veya ağda (Örneğin İnternette) transfer edilen verinin boyunu ölçmede kullanılan birimlerdir.

-Veri Birimleri

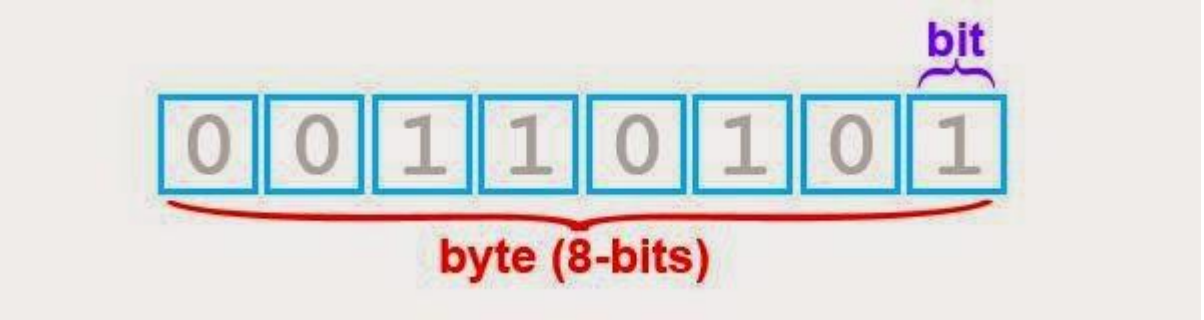
-Ölçü Birimleri

-Hız Birimleri

Bilgisayar Ölçü Birimleri-Nelerdir?

Veri Birimleri

Bit- b : 0 veya 1 değerini alabilen en küçük veri birimi
Nibble- N: 4 bit'ten oluşan veri birimi
Byte-B: 8 bit'ten oluşan veri birimi
Word: 32 bit'ten oluşan veri birimi
Double Word:64 bit'ten oluşan veri birimi
Quad Word: 128 bit'ten oluşan veri birimi
Kilobyte- KB:1024 Byte
Megabyte- MB: 1024 Kilobyte
Gigabyte- GB: 1024 Megabyte
Terabyte- TB: 1024 Gigabyte
Petabyte-PB:1024 Terabyte
Exabyte-EB: 1024 Petabyte
Zettabyte- ZB: 1024 Exabyte
Yottabyte-YB:1024 Zettabyte



Bilgisayar Ölçü Birimleri- Byte Tablosu

METRİK	DEĞER	BAYT
Bayt (B)	1	1
Kilobayt (KB)	$1{,}024^{1}$	1,024
Megabayt (MB)	$1{,}024^{2}$	1.048.576
Gigabyte (GB)	$1{,}024^{3}$	1073741824
Terabayt (TB)	$1{,}024^{4}$	1.099.511.627.776
Petabyte (PB)	$1{,}024^{5}$	1.125.899.906.842.624
Exabyte (EB)	$1{,}024^{6}$	1.152.921.504.606.846.976
Zettabyte (ZB)	$1{,}024^{7}$	1.180.591.620.717.411.303.424
Yottabyte (YB)	$1{,}024^{8}$	1.208.925.819.614.629.174.706.176

Bilgisayar Ölçü Birimleri-Terabyte Nedir?

- Terabayt (TB), sabit sürücü boyutunu ve aslında zaman zaman girebileceğiniz bir sayıyı ölçmek için kullanılan en yaygın birimdir.
- Sadece 1 TB değerinde bilgi saklamak için 728,177 disket veya 1,498 CD-ROM disk çeker.
- 2018 itibariyle, yeni, ortalama fiyatlı bilgisayar sabit diskleri 1 ila 3 TB aralığındadır.
- Birçok ISS, 1 TB'de aylık veri kullanımını kapsamaktadır.
- Hubble Uzay Teleskobu, her yıl yaklaşık 10 TB yeni veri üretir.
- Yaklaşık 130.000 dijital fotoğraf 1 TB alan gerektiriyor.
- IBM'in ünlü Watson oyun oynama süper bilgisayarı 16 TB RAM'e sahip.
- 1 TB'lık veri **bir trilyon bayttan** biraz **daha fazladır** .

1 TB

- **Filmler:** Yaklaşık olarak 1 TB, 200 adet standart kalitede (SD) 2 saatlik filmi depolayabilir. Bu, oldukça büyük bir film koleksiyonu demektir.
- **Yüksek Kaliteli Fotoğraflar:** 1 TB depolama alanı, yüksek kalitede (örneğin, RAW formatında) yaklaşık 200,000'den fazla fotoğrafı saklayabilir.
- **Müzik Dosyaları:** MP3 formatında olan bir şarkı yaklaşık olarak ortalama 5 MB büyüklüğündedir. Bu durumda, 1 TB, yaklaşık 200,000 şarkıyı depolayabilir.
- **Belge ve Metin Dosyaları:** 1 TB, milyonlarca sayfa metin belgesini depolayabilir. Bu, büyük bir kütüphane veya kurumsal belgeler için yeterli olabilir.
- **Yüksek Çözünürlüklü Fotoğraf ve Videolar:** 1 TB depolama alanı, yüksek çözünürlüklü (örneğin, 4K) video dosyalarını ve büyük boyutlu fotoğraf koleksiyonlarını barındırabilir.

Bilgisayar Ölçü Birimleri

Ölçü Birimleri

- **Inch** : 2.54 cm
- **DPI (Dots Per Inch - İnç Başına Nokta):** DPI, bir yazıcı veya görüntüleme cihazının çözünürlüğünü ifade eder. DPI değeri ne kadar yüksekse, görüntü o kadar keskin ve ayrıntılı olur.
- **PPI (Pixels Per Inch - İnç Başına Piksel):** PPI, bir ekran veya görüntüleme cihazının çözünürlüğünü ölçer. Daha yüksek PPI değerleri, daha yüksek çözünürlüklü ekranlar anlamına gelir.
- **Inch (İnç):** İnç, uzunluk ölçü birimidir. Genellikle ekran boyutları, kağıt boyutları gibi büyüklükleri ifade etmek için kullanılır.
- **Pixel (Piksel):** Piksel, bilgisayar ekranlarında ve diğer dijital görüntüleme cihazlarında kullanılan temel renkli noktadır. Çözünürlük genellikle piksel cinsinden ifade edilir.
- **Resolution (Çözünürlük):** Bir ekranın veya görüntüleme cihazının piksel olarak genişlik ve yükseklik ölçüleri.
- **Aspect Ratio (En Boy Oranı):** Bir ekranın genişliği ile yüksekliği arasındaki oran. Genellikle 16:9 veya 4:3 gibi ifade edilir.
- **Bit Depth (Bit Derinliği):** Her bir pikselin rengini ifade eden bit sayısıdır. Daha yüksek bit derinliği, daha fazla renk seçeneği sağlar.
- **Refresh Rate (Yenileme Hızı):** Ekranın saniyede kaç kez yenilendiğini ifade eder. Yüksek yenileme hızları daha akıcı bir görüntü sağlar.

DPI (Dots Per Inch - İnç Başına Nokta)

- **Düşük DPI (72 DPI) Örneği:**

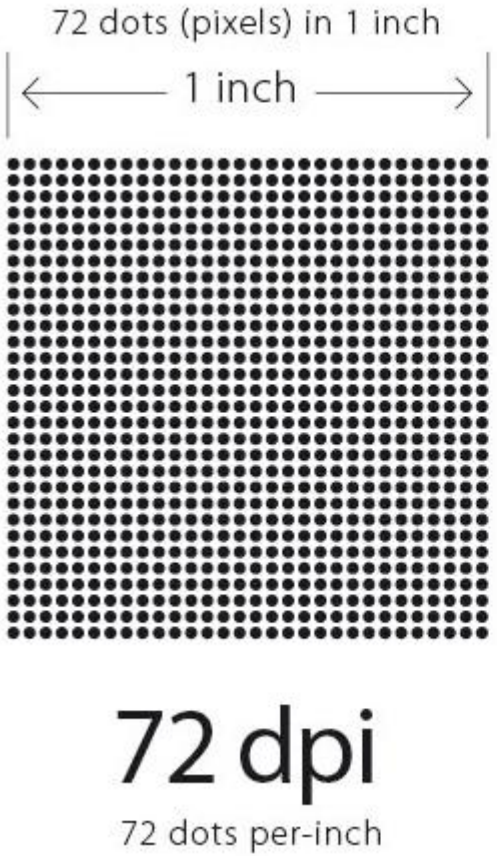
Eğer bir resmin DPI değeri düşükse, örneğin 72 DPI, resim daha az piksel içerecektir her bir inç başına. Bu, düşük kaliteli bir görüntüye neden olabilir, özellikle de baskıya dönüştürüldüğünde. Ayrıntılar kaybolabilir ve pikselleşmiş bir görüntü ortaya çıkabilir.

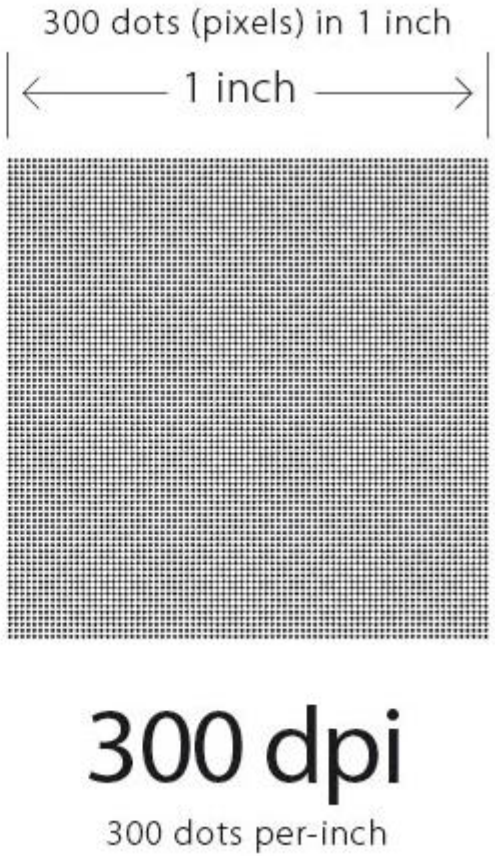
- **Orta DPI (150-200 DPI) Örneği:**

Daha yaygın olarak kullanılan bir DPI değeri 150 ila 200 arasında olan bir resim, daha iyi kalitede bir görüntü sağlayacaktır. Bu değerler, genellikle çoğu ofis yazıcısının kullanabileceği kalitede baskılar üretmek için yeterlidir.

- **Yüksek DPI (300 DPI ve üstü) Örneği:**

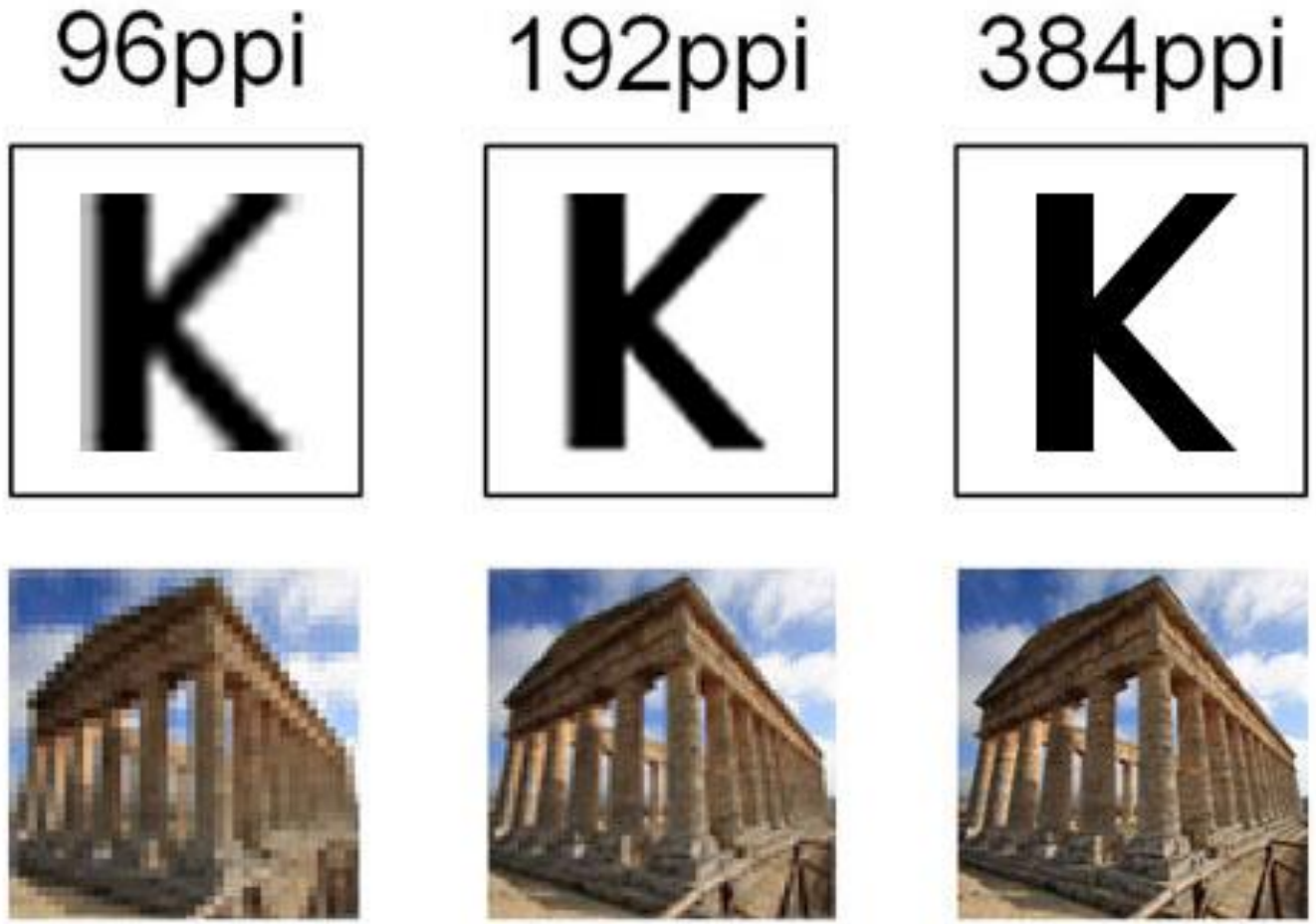
Profesyonel baskı projeleri için, genellikle 300 DPI veya daha yüksek DPI değerlerine ihtiyaç vardır. Bu yüksek DPI değeri, baskıda çok daha keskin ve ayrıntılı bir görüntü sağlar. Fotoğraflar, afişler ve diğer profesyonel baskı projeleri genellikle bu yüksek DPI değerlerinde oluşturulur.



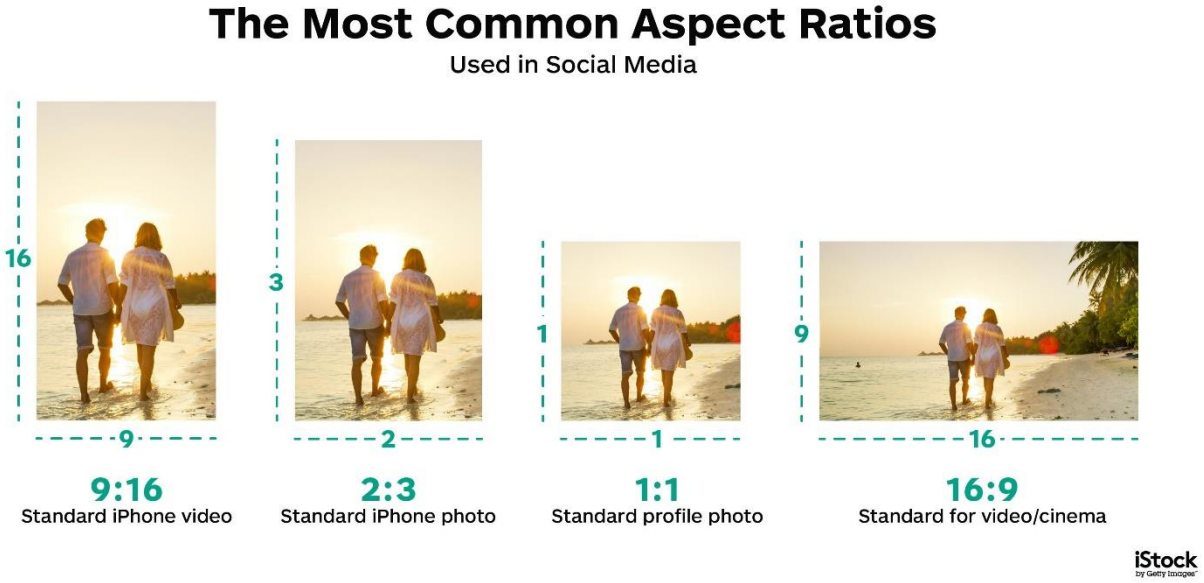


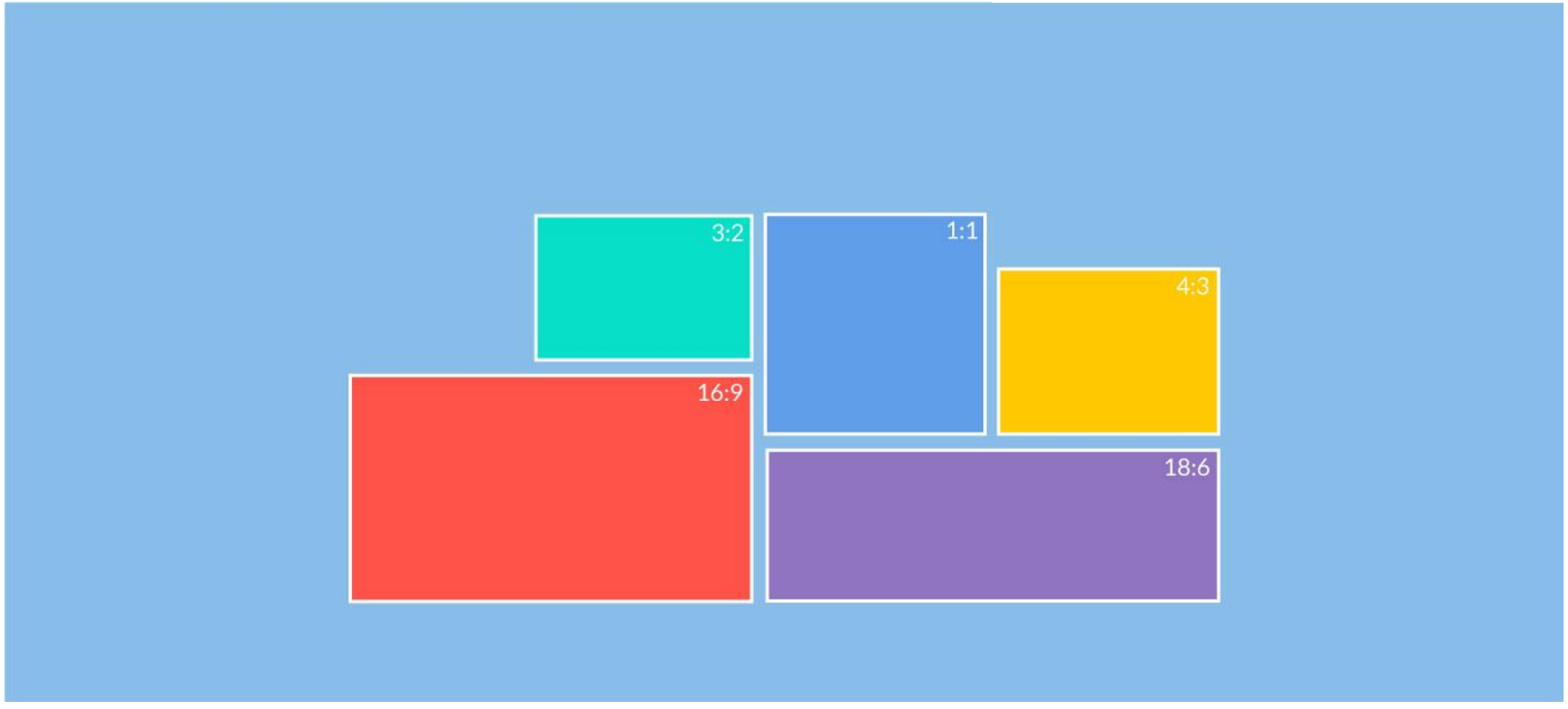
PPI

- PPI, özellikle dijital ekranlardaki görüntü kalitesini değerlendirmek için önemlidir. Daha yüksek PPI değerleri, daha yüksek çözünürlüklü ve daha keskin görüntüler sağlar. Örneğin, aynı boyutta iki farklı ekrandan biri 300 PPI, diğeri 150 PPI değerine sahipse, 300 PPI değerine sahip ekran daha keskin ve daha ayrıntılı görünecektir.

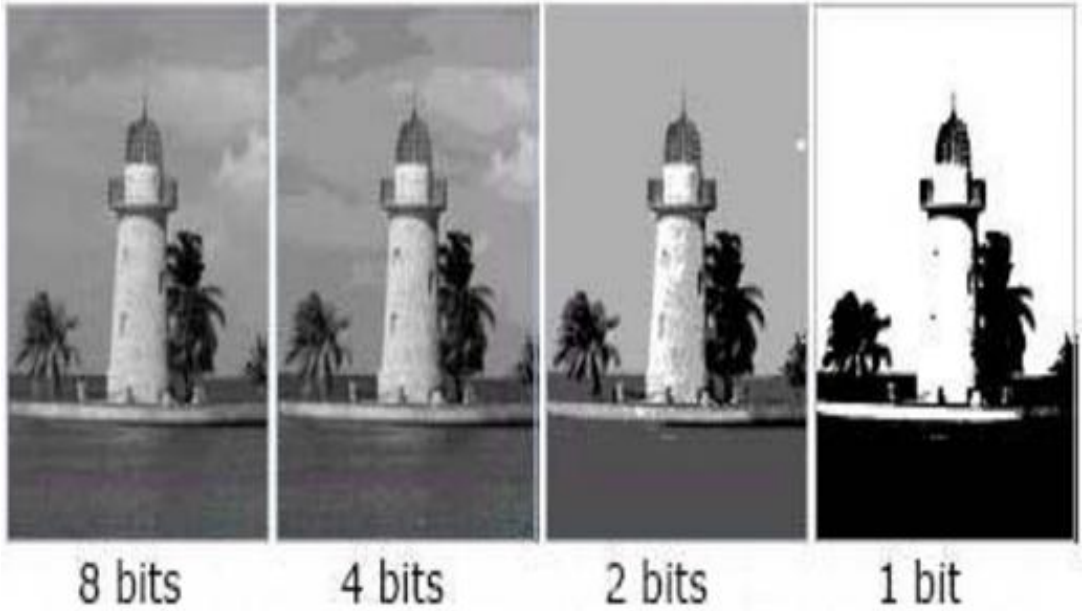


Aspect Ratio



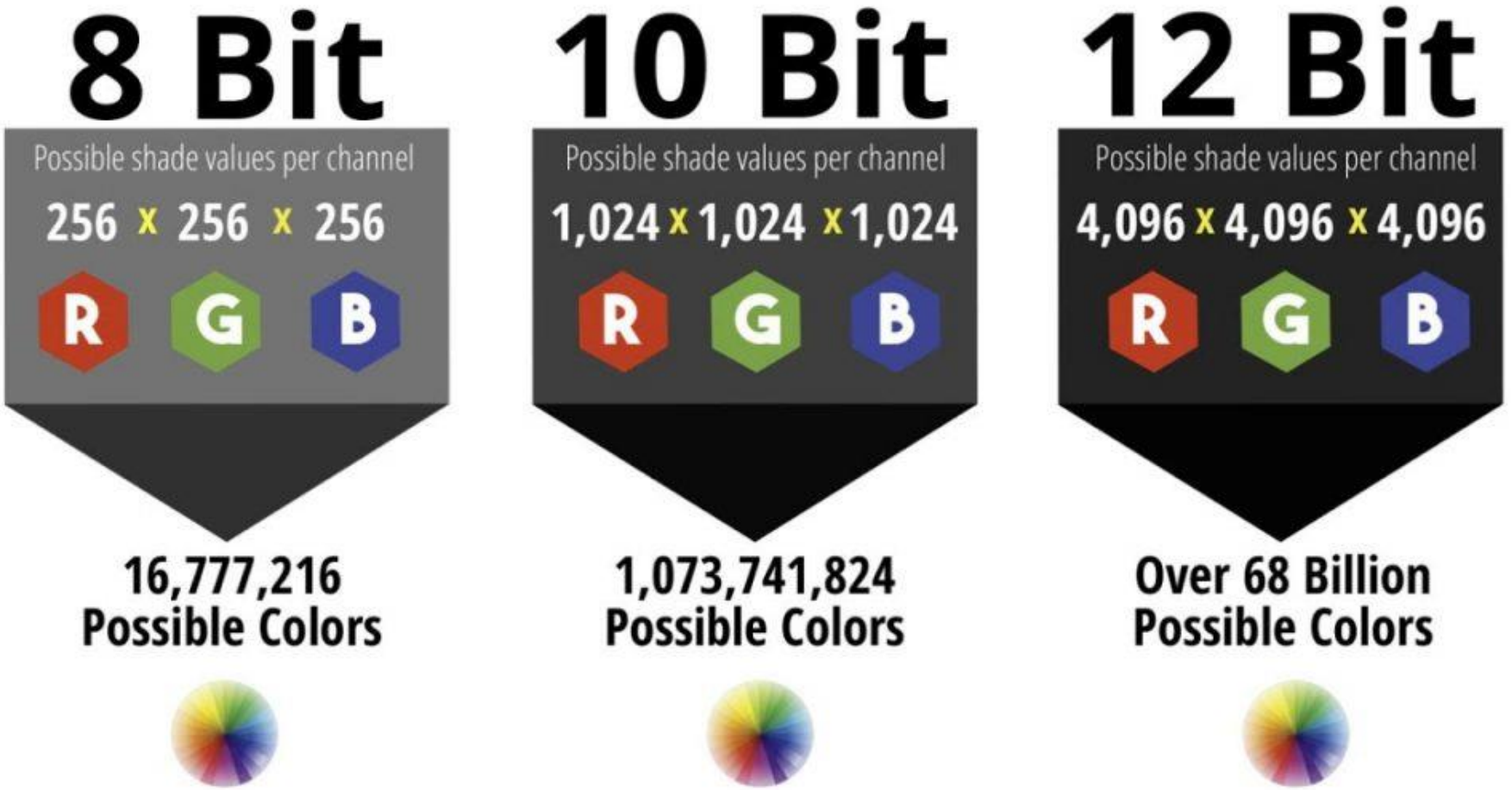


Bit Derinliği

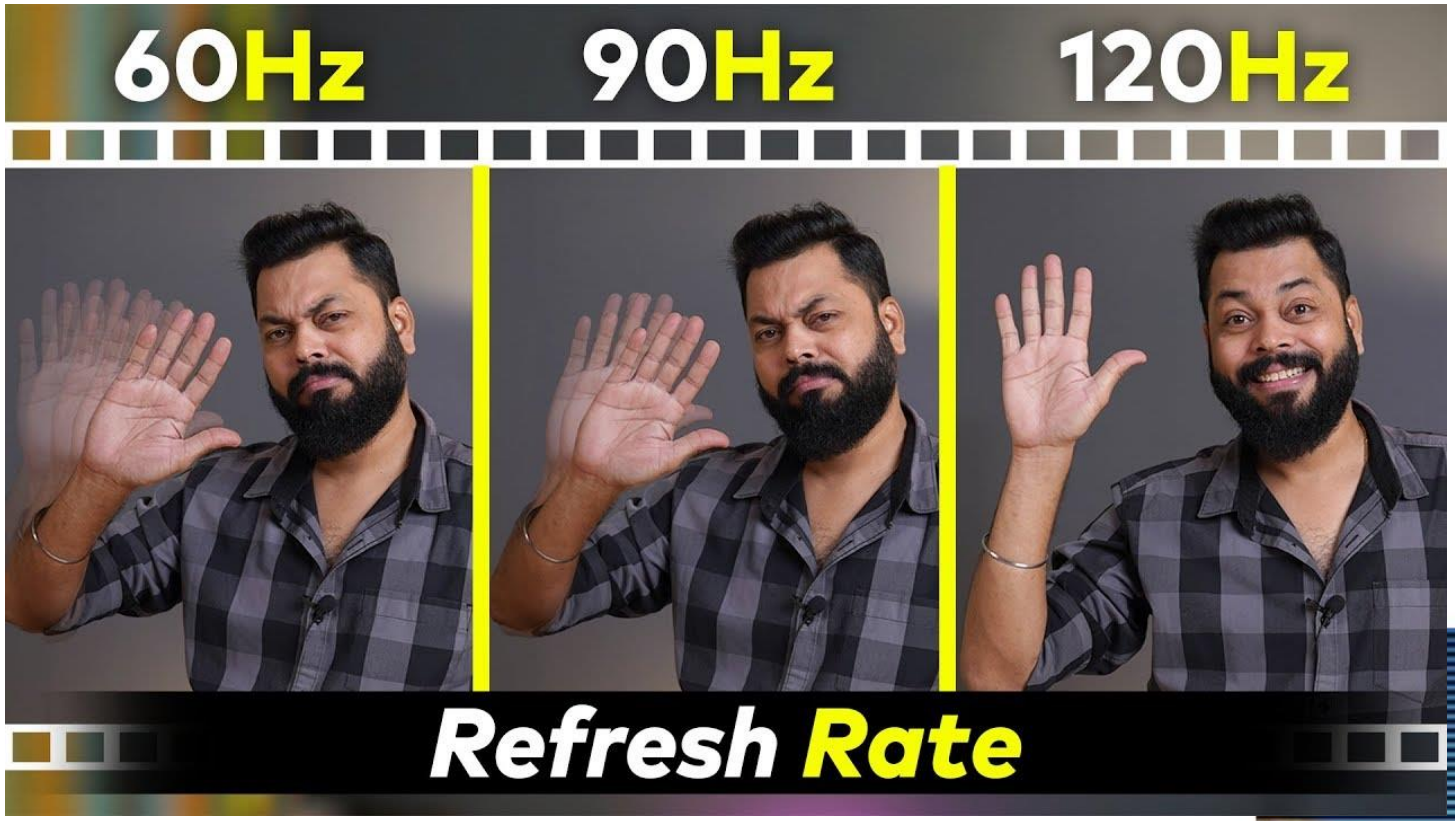


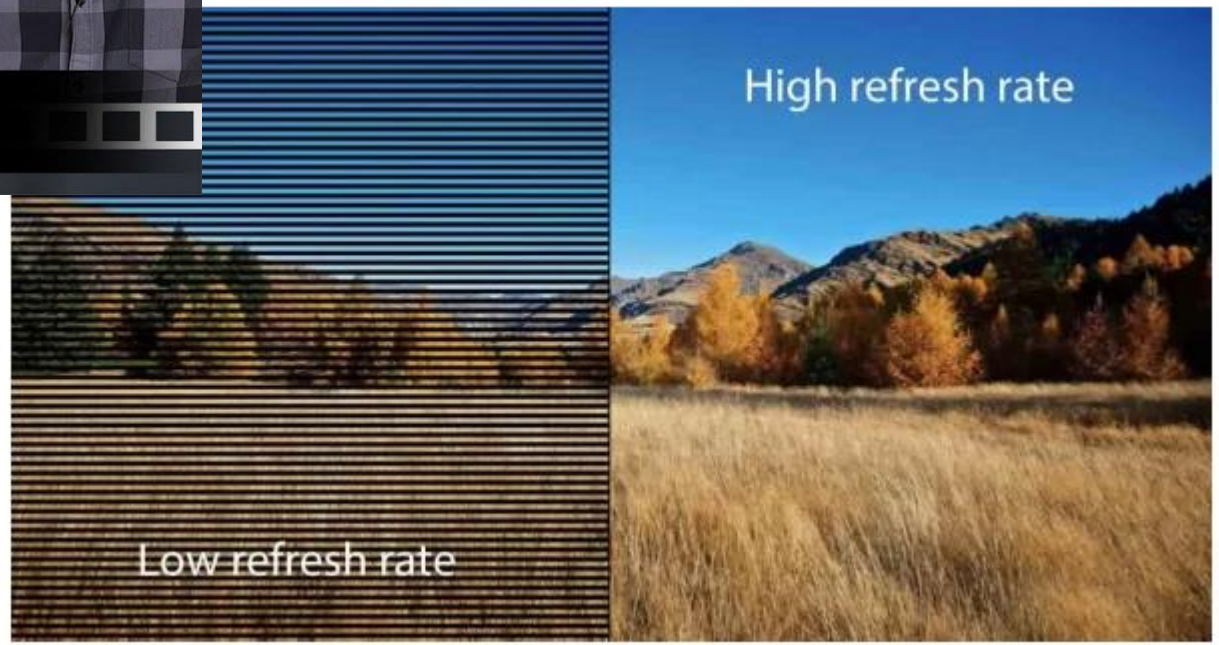
- **Örnek:**
- Bir 8-bit renk derinliğine sahip bir görüntüde (256 renk tonu), her piksel 0 ile 255 arasında bir değeri temsil edebilir. Örneğin, bir pikselin değeri 0 ise siyah, 255 ise beyaz olabilir. Diğer değerler arasında gri tonları veya diğer renk tonları bulunabilir.
- Daha karmaşık bir örnek olarak, bir 24-bit renk derinliğine sahip bir görüntüde (16.7 milyon renk tonu), her biri kırmızı, yeşil ve mavi (RGB) bileşenlerinden oluşan 24 bitlik bir değeri temsil edebilir. Örneğin, 0xFF0000 değeri kırmızı rengi temsil ederken, 0x00FF00 yeşil ve 0x0000FF maviyi temsil eder. Bu renklerin farklı kombinasyonları, milyonlarca farklı renk tonunu oluşturur.

Bit Derinliği



Refresh Rate





refresh rate comparison

Bilgisayar Ölçü Birimleri
Hız Birimleri

- Hertz - Hz : Bir saniyedeki devir/yenileme/tekrar sayısı.
- Kilohertz – kHz: Saniyedeki 1000 Hertz'den oluşan hız birimi.
- Megahertz – MHz : Saniyedeki 1000 kHz'den oluşan hız birimi.
- Gigahertz – GHz: Saniyedeki 1000 MHz'den oluşan hız birimi.
- Flops: (floating point operations per second) saniyedeki reel sayılı işlem sayısı.
- Mflops: saniyedeki1000.000 Flops tan oluşan hız birimi
- GigaFlops: saniyedeki 1000 Mflopstan oluşan hız birimi
- bps: bits per second. Saniyedeki iletilen bit sayısı
- Bps: Bytes per second. Saniye iletilen byte sayısı.
- MIPS: Millions Instructions Per Second: saniyede milyon cinsinden komut sayısı
- Revolutions per minute: RPM: dakikadaki devir sayısı- HDD
- Bit rate: iletilen bit sayısı bit oranı
- Refresh rate: saniyedeki görüntü yenileme hızı



- Ampullerin yanıp sönme hızı, genellikle "yanıp sönme frekansı" olarak adlandırılır ve bu değer genellikle çok yüksektir, bu nedenle insan gözü tarafından algılanamaz. Standart bir elektrik sisteminde (örneğin, 50 Hz frekansında çalışan bir AC sistemi), ampuller ortalama olarak 100 kez (yanıp sönmeye) 120 kez (yanıp sönme döngüsüne) kadar her saniyede yanıp söner. Bu döngü hızı insan gözü tarafından algılanamayacak kadar hızlıdır ve ampul sürekli bir şekilde yanıyor gibi görünür. Bu nedenle, ampul dakikada birçok kez yanıp söner, ancak bu hız insan gözü tarafından algılanamaz.

Temel Kavramlar-

- Bilgisayar/Yazılım Mühendisliği
- Bilgisayar Organizasyonu: Merkezi İşlem Birimi, Bellek ve Saklama Birimleri, Bellek hiyerarşisi, Giriş Çıkış Birimleri
- Bilgisayar Ağları
- Bilgisayar Yazılımı, Kod, Algoritma, Veri, Program
- İşletim Sistemleri

Bilgisayar/Yazılım Mühendisliği Eğitimi ve Görevleri

- Mühendis: Teorik bilgi → Pratik, Hendese & Matematik
- Mühendislik: üretim, işletme, bakım
- Yazılım mühendisi: Mühendisliğin sistematik olarak yazılım alanına uygulanması.
- Programcı(lık)!
- Programlama? / Programming, Implementation
- Programlanacak sistemlerin tasarımı,
- Platform, teknoloji ve metodolojilerin geliştirilmesi,
- Algoritmik, analitik düşünme yeteneği,

Bilgisayar/Yazılım Mühendisliği Eğitimi ve Görevleri

Tanım

- Bilgisayar sistemlerinin kullanıldığı bankacılık, otomotiv, telekomünikasyon, enerji, iletişim, vb. gibi sektörlerin ve kamu kuruluşlarının yazılım tasarımını hazırlayan ve sisteme entegre edip uygulanması,

Görevler

- Hazırlanacak yazılım ile ilgili **müşteri ilişkileri**ni kurar, Müşterilerden aldığı bilgiler doğrultusunda hazırlanacak yazılımın **analiz**ini yapar, **tasarım**ını yapar, programını yazar(**implementation**), programını **test** eder, yazdığı programda eksiklik varsa düzeltir, yazılımı sorunsuz şekilde çalışır hale getirdikten sonra, gerekli durumlarda kullanım yönergesini hazırlayarak, kullanıcıların eğitilmesini sağlar, mesleği ile ilgili gelişmeleri takip eder.

Bilgisayar/Yazılım Mühendisliği Eğitimi ve Görevleri

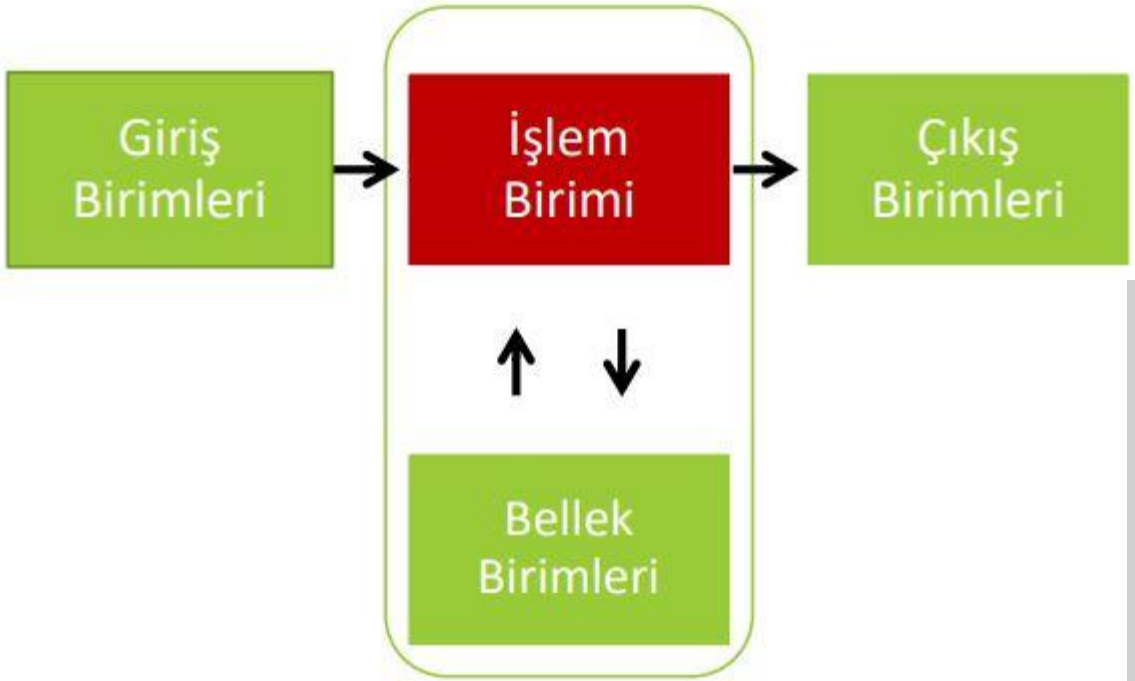
Mesleğin Gerektirdiği Özellikler

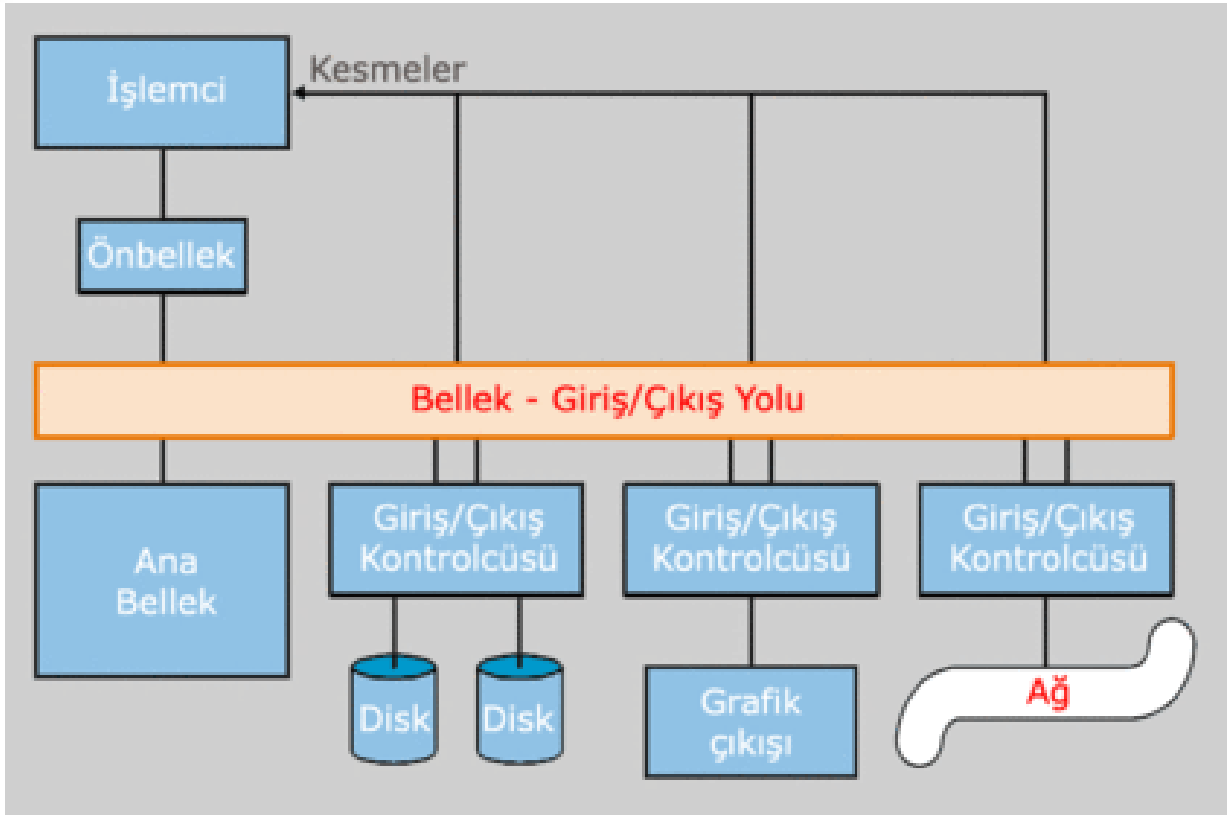
- Yazılım mühendisi olmak isteyenlerin; **mantık** yürütme ve **tasarım** yeteneği olan, **matematik** alanlarına ilgi duyan, bildiklerini başkalarına aktarabilme ve etkileme (ikna) yeteneği olan, dikkatli, sabırlı, **ekip çalışması**na ayak uydurabilen, yeni fikirlere açık (**innovation**), kendisini sürekli geliştirebilen kimseler olması gerekir.

Bilgisayar/Yazılım Mühendisliği Konuları

- Bilgisayarın matematiği, Ayrık Matematik
- Veri yapıları ve algoritmalar
- Veritabanları ve ilişkisel veri modeli
- İşletim sistemleri
- Mikroişlemciler ve gömülü sistemler
- Yazılım mühendisliği ve yazılım modelleme dilleri
- Algoritma analizi ve optimizasyon
- Nesneye yönelik programlama
- Web teknolojileri
- Yazılım tasarım ve mimarisi
- Yazılım kalite, doğrulama, geçerleme, test
- Yazılım mühendisliğindeki güncel gelişmeler

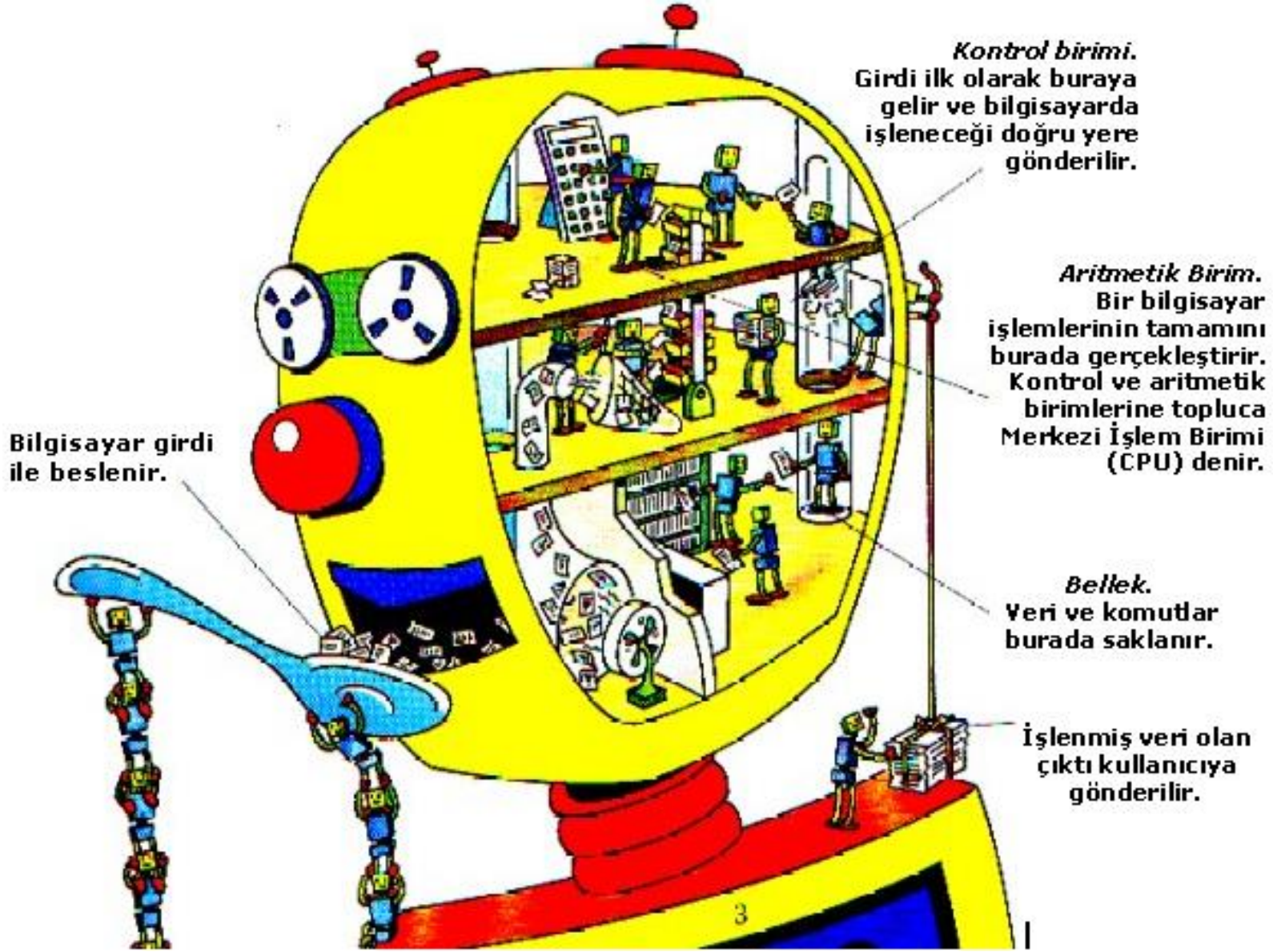
Bilgisayar Donanımı-





Bilgisayar Donanımı

- Merkezi İşlem Birimi (CPU)
- Ana Bellek (RAM)
- Ön Bellek (Cache Memory)
- BIOS (Basic Input Output System)
- Anakart-Motherboard
- Bilgisayar Kasası- Tower
- Sabit Disk (Hard disk-HDD)
- CD-ROM, DVD-ROM Sürücü
- Monitör (Ekran)
- Ekran Kartı-GPU
- Klavye
- Mouse (Fare)
- Yazıcı (printer)
- Çizici (plotter)
- Tarayıcı (scanner)
- Modem
- NIC

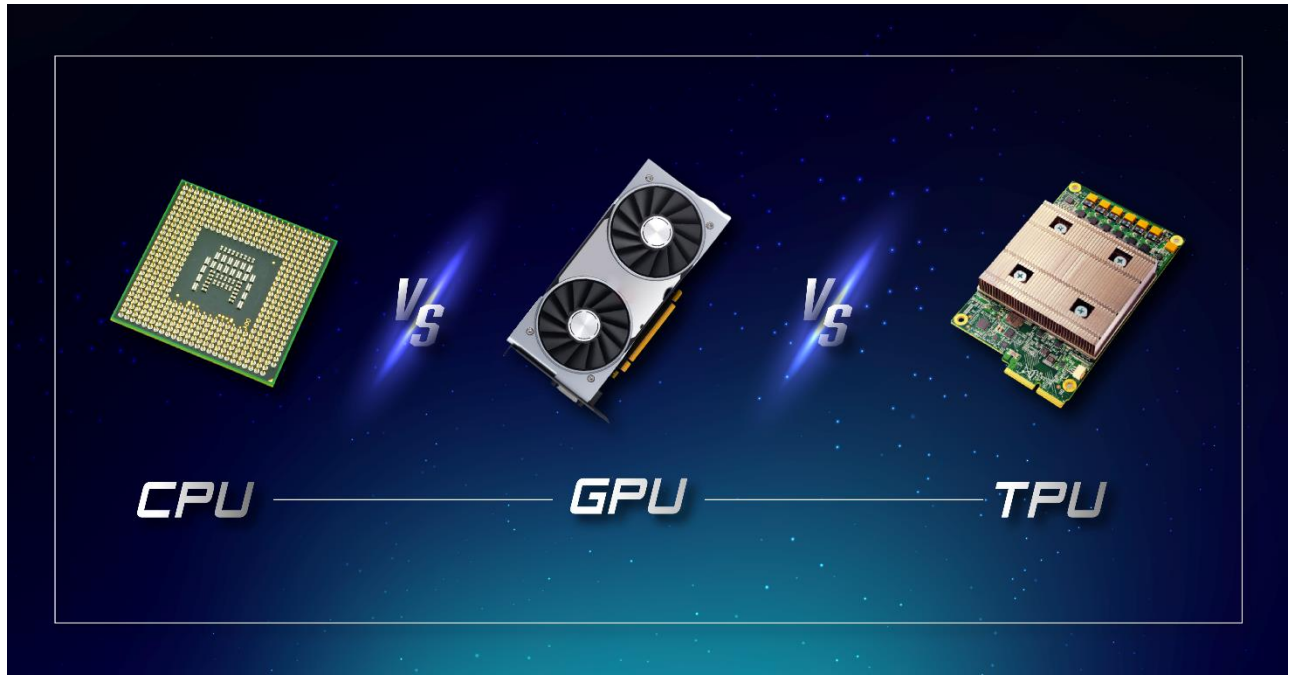


Bilgisayar Donanımı-CPU

- CPU-Central Processing Unit- Merkezi işlem birimi
- Bilgisayarın beyni
- Cache
- Hertz
- Komutları işler
- Hesaplamaları yapar-Aritmetic logic unit- ALU
- Bilgisayar içindeki bilgi akışını kontrol eder ve yönetir
- Makine dili, makine kodu 0xFB, assembly* add,
- *her işlemci ailesinin kendine has assembly dili vardır!
- RISC, CISC, von neuman ve Harvard mimari yaklaşımları



CPU, GPU, TPU



CPU
- Small models
- Small datasets
- Useful for design space exploration

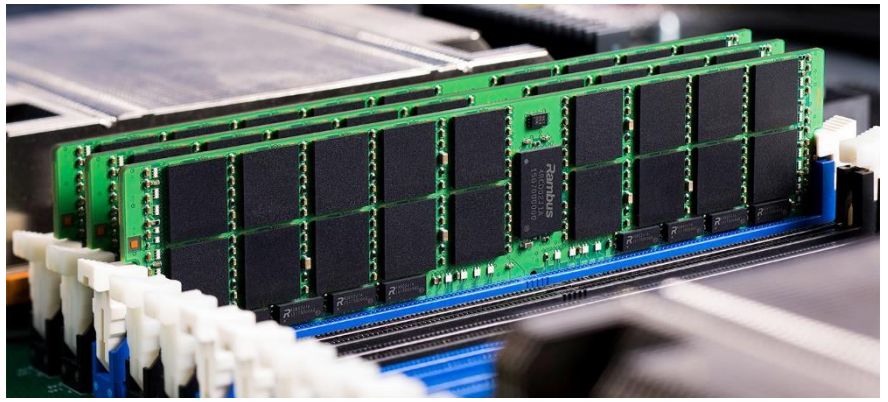
GPU
- Medium-to-large models, datasets
- Image, video processing
- Application on CUDA or OpenCL

TPU
- Matrix computations
- Dense vector processing
- No custom TensorFlow operations

FPGA
- Large datasets, models
- Compute intensive applications
- High performance, high perf./cost ratio

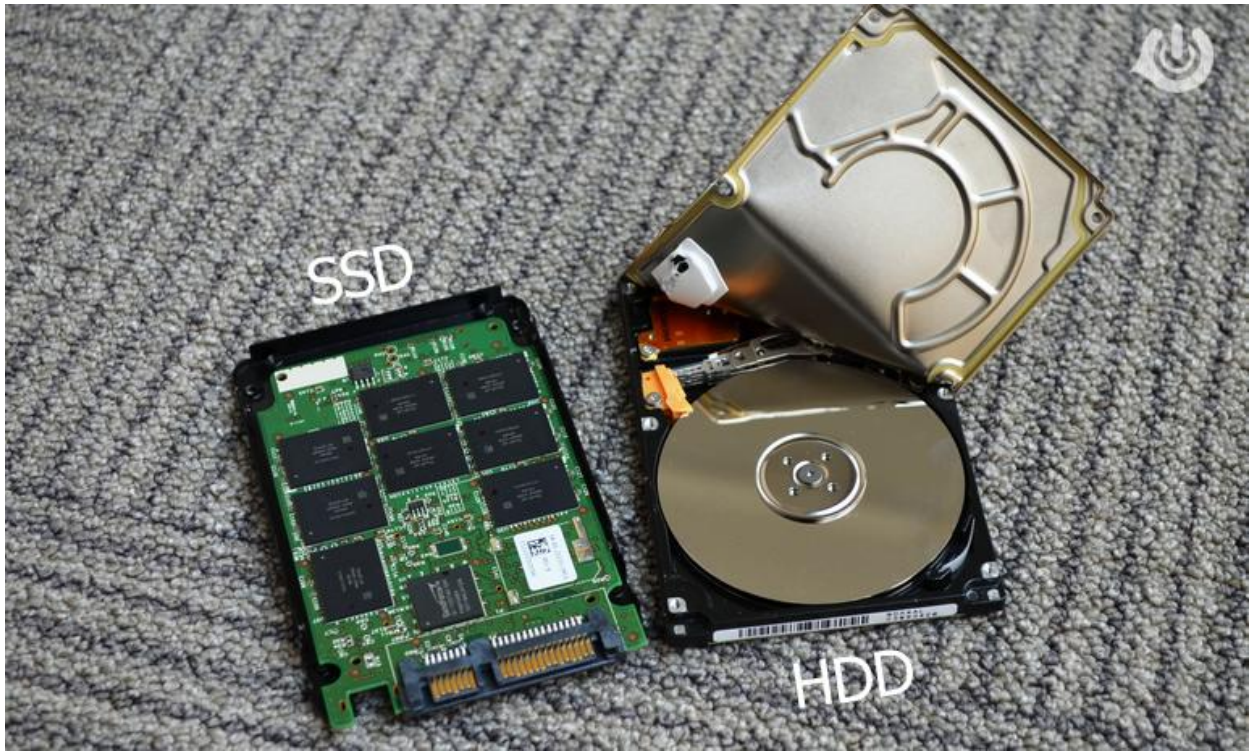
Bilgisayar Donanımı-Hafıza-Bellek-Depolama

- CPU'lar üzerlerinde hafıza birimlerine sahiptir ancak bunlar küçük ve geçicidir
  - Registerlar- cpu üzerindeki kayıtçılardır
  - Cache bellek-
  - Volatile – Geçici
- Daha büyük kapasite için RAM, ROM
  - RAM(Random Access Memory), Ana bellek,
  - ROM(Read Only Memory), (PROM, EPROM, EEPROM)
- Daha kalıcı hafıza için depolama birimleri
  - Sabit Disk-HDD, Disket sürücü, USB Bellek vb.
  - CD/DVD ROM
  - Non-volatile - Kalıcı



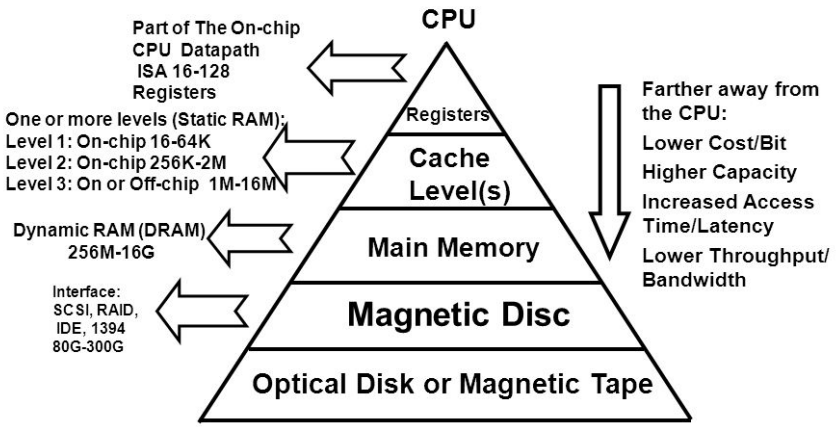


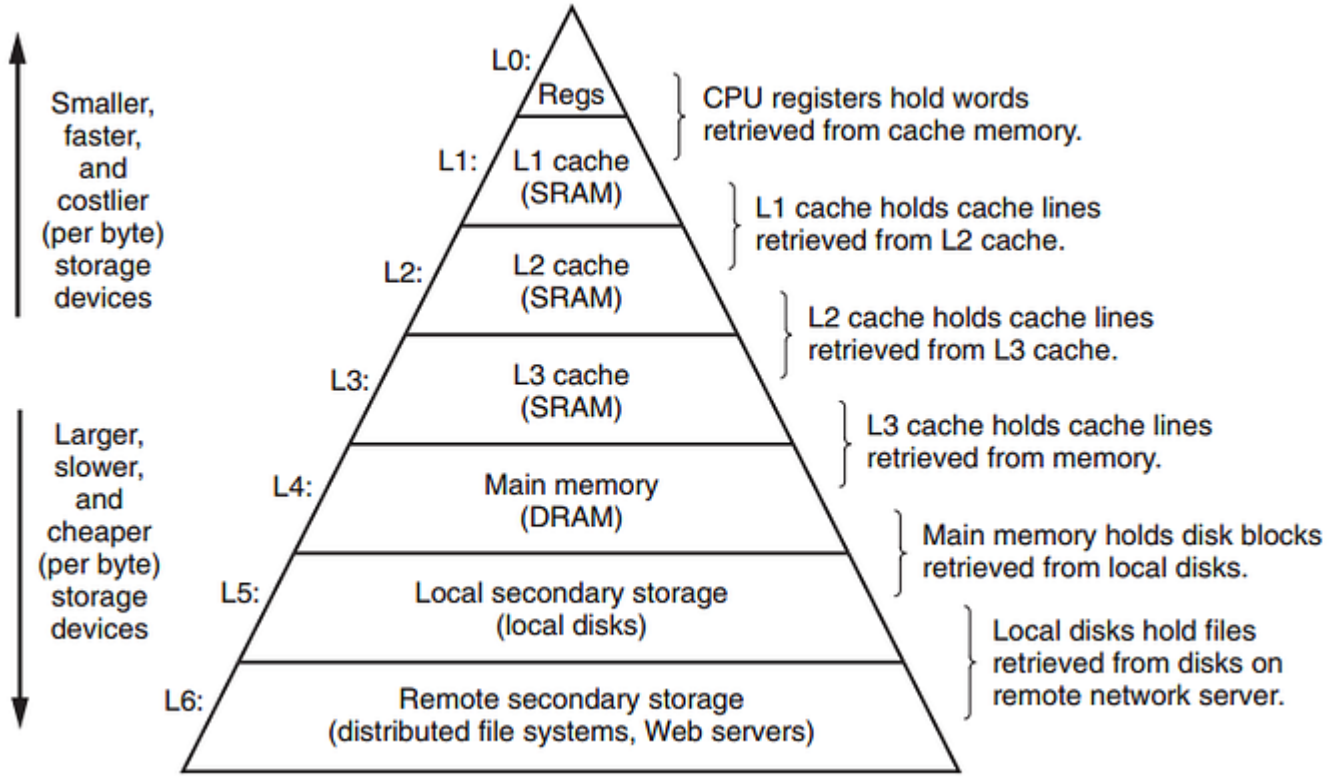
HDD, SSD


SSD
HDD


Mil
Okuma/Yazma Kafası
Kafa Taşıyıcı Kol
Mil
Kafa Hareketlendiricisi
Kasa Bağlantı Noktaları
Disk Plakaları
Veri Kablosu

Bilgisayar Donanımı-Bellek Hiyerarşisi





Bilgisayar Donanımı- Giriş/Çıkış I/O Üniteleri

- Monitör, ekran kartı, klavye, fare…









Plotter, Printer











Bilgisayar Donanımı- Ağ donanımları

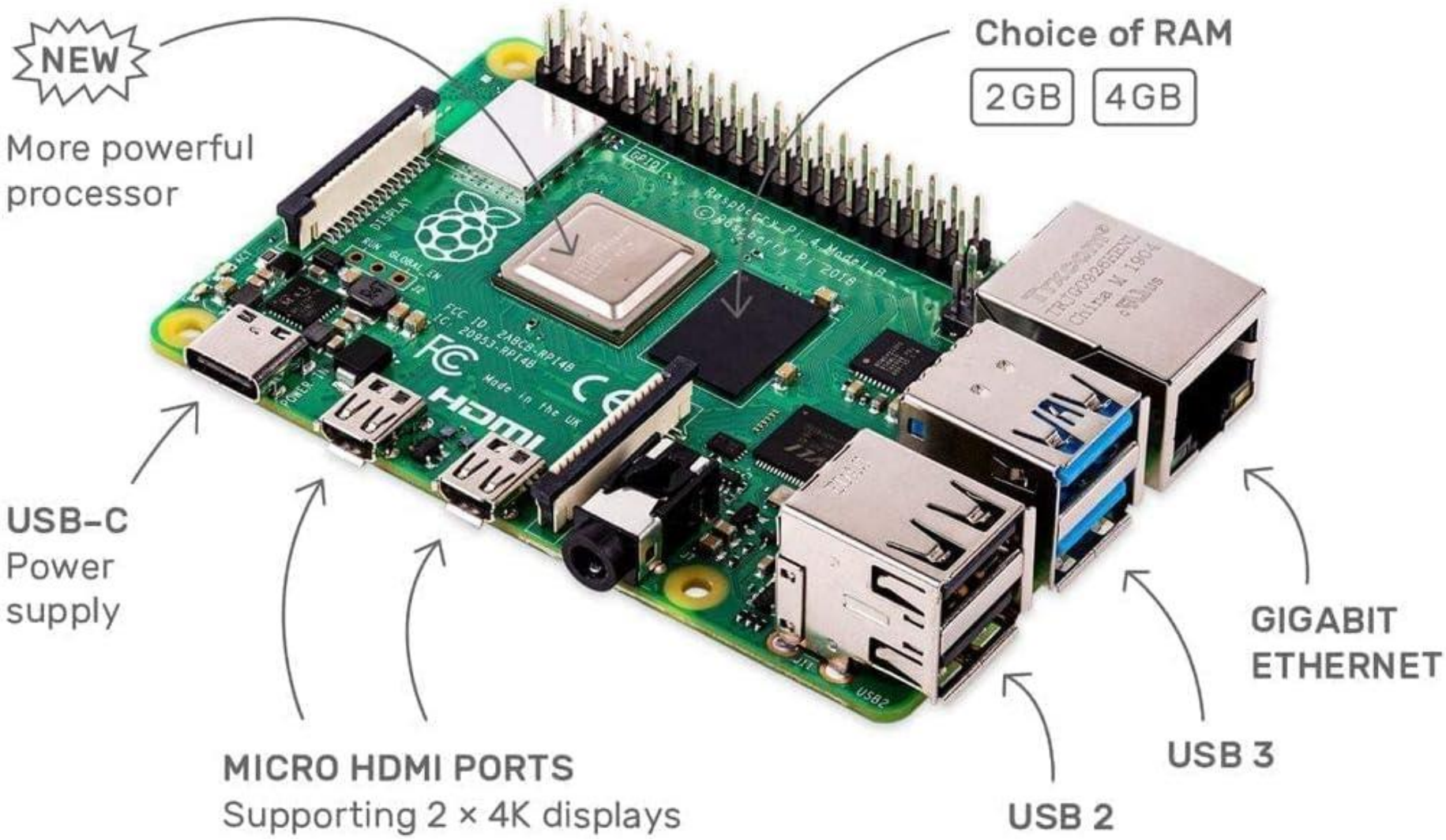
- Ethernet kartı-NIC- Anakarta entegre
- Modem, **Mo**dulator, **dem**odulator
- Switch
- Router
- Gateway
- Firewall







Marka	Raspberry Pi
Üretici	Raspberry Pi
Ürün Boyutları	9.5 x 7 x 2.8 cm; 50 gram
Piller:	1 Bilinmeyen pil gereklidir.
Ürün Model Numarası	102110421
İşlemci Yuvası	Socket3
İşlemci Sayısı	1
RAM Boyutu	8 GB
Bellek Teknolojisi	DDR3
Bilgisayar Bellek Türü	DDR3 SDRAM
Desteklenen Maksimum Bellek	8 GB
Bellek Saat Hızı	2.4 GHz
Ekran Kartı Arayüzü	PCI-Express x4
Bağlantı Türü	Bluetooth
Kablosuz Türü	Bluetooth
Voltaj	5 Volts
Piller Dahil mi?	Hayır
Lityum Metal Hücre Sayısı	1
Üretici	Raspberry Pi
ASIN	B0899VXM8F
Ürün Ağırlığı	50 g



RAIDER GE78HX 13VH-038TR



•İşlemci: Intel® Core™ i7-13700HX (30M Cache, up to 5.00 GHz)
•İşletim Sistemi: Windows® 11 Home Advanced
•Ekran: 17" 16:10 QHD+(2560 x 1600), 240Hz, 100% DCI-P3
•Chipset: Intel HM770
•Ekran Kartı: RTX 4080, GDDR6 12GB
•Hafıza: DDR V 32GB (16GB*2, 4800MHz)
•Hafıza yuvası: 2 Slot
•Maksimum Hafıza: Max 64GB
•HDD: 2TB NVMe SSD
•Depolama Kapasitesi: "1x M.2 SSD slot (NVMe PCIe Gen4) 1x M.2 SSD slot (NVMe PCIe Gen5) Compatible"
•Ön Kamera: IR FHD type (30fps@1080p)
•Klavye: Per-Key RGB Keyboard by SteelSeries
•Pil: 4-Cells, Li-Polymer, 99 Whr
•Güç Adaptörü: 330W
•Boyutlar: 380 x 298 x 23 mm
•Ağırlık: 3.1 kg

İşletim Sistemi

- Yönetici, çatı yazılım
- DOS, Windows, Linux, Unix, Android, IOS, Pardus, GIBUX
- Çok kullanıcılı, çoklu görev yönetimi,
- Multitasking!? Multithreading





Algoritma, veri yapıları, programlama dilleri

- Algoritma
  - Sonlu sıralı sayıda adım, yöntem, yol
  - Genel, etkin, kesin bir metodoloji
  - Girdi, veri işleme/analiz, çıkış üretme
- Algoritmaların ifade edilme şekilleri
  - Akış şemaları
  - Sözde/Kaba kod, pseudocode
  - Sözel anlatım-Metinsel ifade
- Veri yapıları: verilerin bilgilere dönüştürülerek bellekte tutulma şekil ve düzenini gösteren yapılardır.
- Programlama dilleri: Sürekli gelişen güçlü programlama ortam ve araçları

Algoritma, veri yapıları, programlama dilleri

- Veritabanları: Birbiri ile ilişkili verilerin tutulduğu yapılar
  - Depolama
  - Yetkilendirme
  - Sorgulama
  - RDBMS-İlişkisel veritabanı yönetim sistemi
  - SQL-Yapısal sorgulama dili
  - Oracle, MSSQL, sybase, PostgreSQL
  - NoSQL

Veri Güvenliği

- Bilgi güvenliği, siber güvenlik
- Gizlilik(Con. Cry.), Bütünlük(Int. Öz.), Erişilebilirlik (Ava. MTTF vb.), Yetkilendirme (Aut.), İnkar edilemezlik (Non rep. E-sig.)
- Siber güvenlik, istihdam gereksinimi
- ISO 27001, UEKAE BGYS 0001,
- 5651, 6698 yasalar
- NSA, NIST 800-026- Self-assesment

Sayı Sistemleri

Desimal Sistem

- Günlük hayatımızda desimal sistemi kullanmaktayız
- Tabanı 10'dur

Örn:

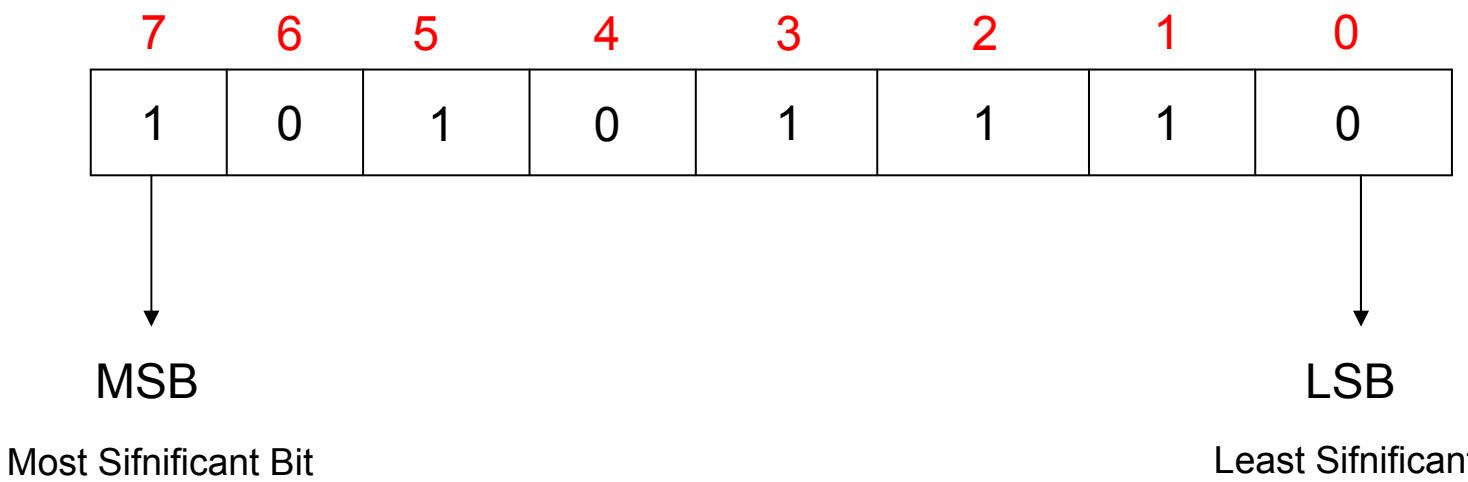
$$365 = 3.10^2+6.10^1+5.10^0$$

$$4827 = 4.10^3+8.10^2+2.10^1+7.10^0$$

İkili Sayı Sistemi (Binary System)

- Sistemin Tabanı 2'dir.
- Sadece "0" ve "1" kullanır.
- Her sayı "dijit" olarak ifade edilir.
- Basamaklar 2'nin kuvveti olarak yazılır.

Bit Etiketleri



Binary-Desimal Çevrimi

- Her dijit 2'nin kuvveti ile çarpılarak desimal sayı sistemine çevrilir.

Örn:

i) $(101011)_2$

$=1.2^5+0.2^4+1.2^3+0.2^2+1.2^1+1.2^0 =43$

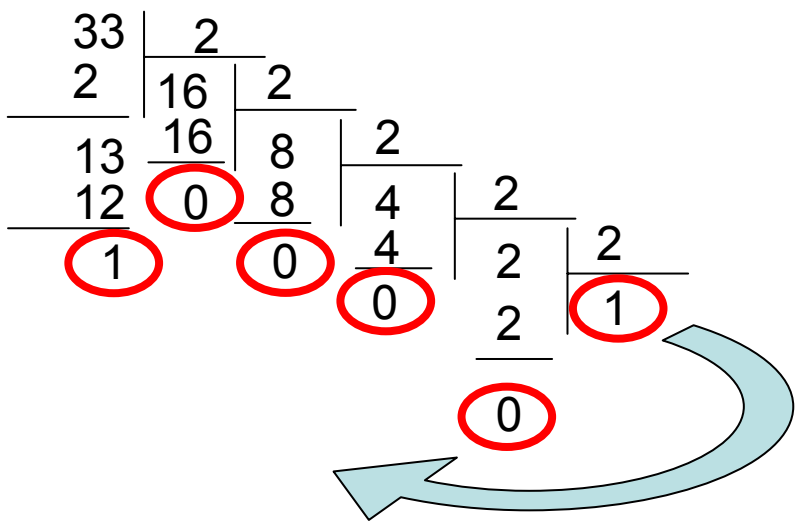
ii) $(1011)_2$

$=1.2^3+0.2^2+1.2^1+1.2^0 =11$

Desimal-Binary Çevrimi

- Desimal sayıları binary sayılara çevirirken “Bölme-2” metodu kullanılır. Çıkan sonuç tersten yazılır.

  Örn: $(33)_{10}=(?)_2$



$(33)_{10}=(100001)_2$

Tam ve Ondalıklı Binary Sayıların Desimal Sayılara Çevrimi

- İlk önce binary sayı 2'nin kuvveti olarak yazılır
  - Binary sayının tam kısmı pozitif 2'nin kuvveti olarak yazılır
  - Ondalıklı kısmı ise, soldaki ilk dijitten başlayarak sağa doğru negatif 2'nin kuvveti olarak yazılır.
- Ondalıklı kısmın 2'nin kuvvetleri bayağı kesirli hale getirilir ve kesirlerin desimal değerleri bulunarak dijitlerle çarpılır.
- Çıkan sonuç toplanarak desimal sayı elde edilir.

Örn:

$(100,10001)_2=(?)_{10}$

$=1.2^2+0.2^1+0.2^0+1.2^{-1}+0.2^{-2}+0.2^{-3}+0.2^{-4}+1.2^5$

$=4+1/2+1/2^{32}=4,50000000023$

Tam ve Ondalıklı Desimal Sayıların Binary Sayılara Çevrilmesi

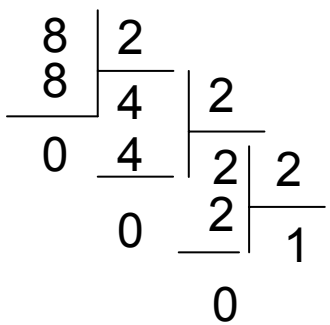
- 2 aşamada gerçekleşir
- İlk önce tamsayı “Bölme 2” metodu ile binarye çevrilir.
- Daha sonra ondalıklı sayı “Çarpma2” metodu ile binarye çevrilir.

Not: Ondalıklı kısım, 1,00 sayısı elde edilene kadar sürekli 2 ile çarpılır. Eğer 1,00 sayısı elde edilmeden sonlandırılırsa gerçek sayı elde edilemez.

Örn:

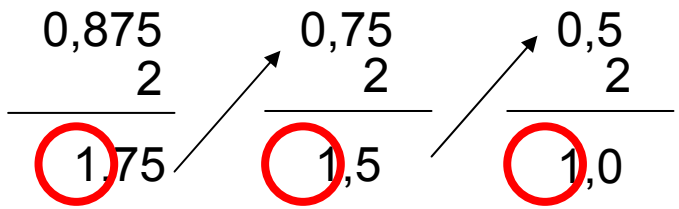
$(8,875)_{10}=(?)_2$

i)



$(8)_{10}=(1000)_2$

ii)



$(8,875)_{10}=(1000,111)_2$

Binary Sayı Sisteminde 4 İşlem

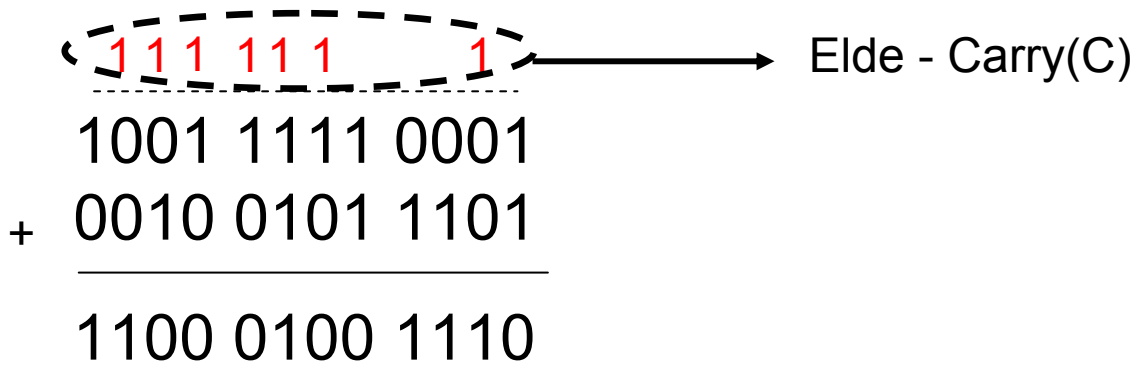
-A-Toplama

Kurallar:

i) 0+0=0	ii) 0+1=1	iii)1+1=10	iv)1+1+1=11	v)1+1+1+1=100

Not: İkinci basamağa geçen “1” sayısı “Carry (C)” ile gösterilir.

Örn:


1 1 1 1 1 1 1
Elde - Carry(C)
1001 1111 0001
+ 0010 0101 1101
1100 0100 1110

-B-Çıkartma

Kurallar:

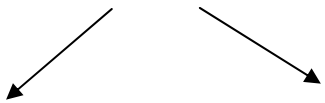
i) 0-0=0 ii) 1-1=0 iii)1-0=1 iv)0-1=1 → X

Not: X üst basamaktan alınan borç (borrow-b) ile ifade edilir.

```
   x    11x  xx
 ----------------------------
  1101 1001 1001
- 1010 0101 1101
 ----------------------------
  0011 0011 1100
```

Tamamlayıcı (Complementer) Yoluyla Çıkartma İşlemi

2 tür tamamlayıcı vardır



1 tamamlayıcısı 2 tamamlayıcısı

Not :

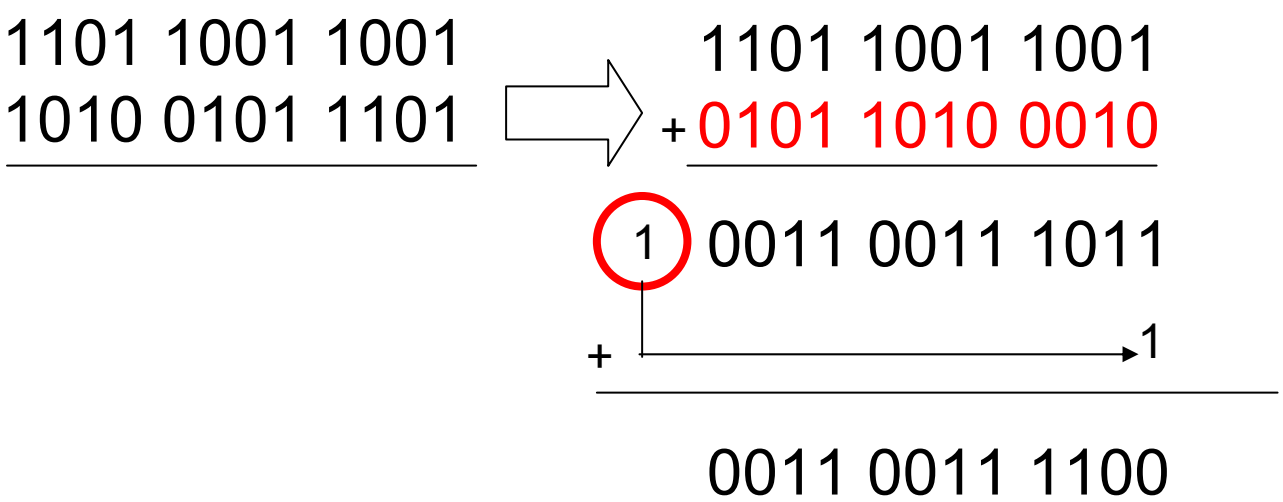
i) $(XXXX)_2$ sayısının 1 tamamlayıcısı $(1111)_2$ sayısından çıkarılarak elde edilir.

ii) Pratik olarak bir binary sayının 1 tamamlayıcısı dijitlerinin tersi alınarak bulunur. (0→1; 1→0)

iii) Sayının 2 tamamlayıcısı = Sayının 1 tamamlayıcısı + 1'dir.

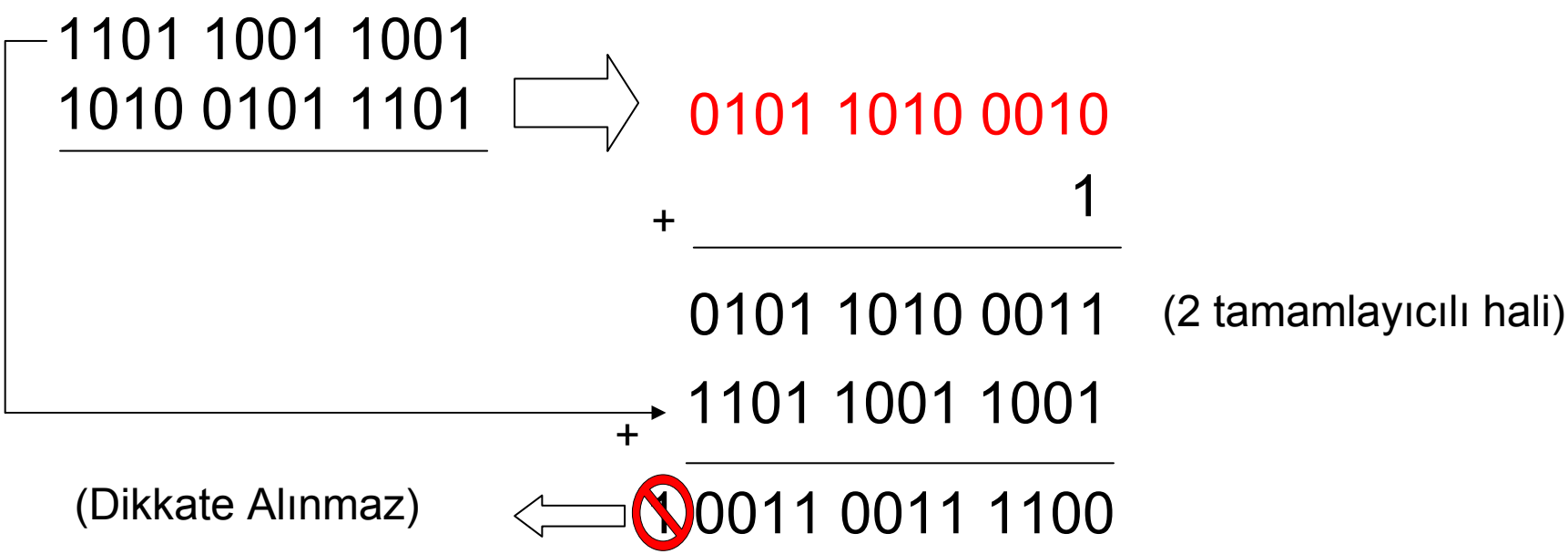
1 Tamamlayıcısı Yardımıyla Çıkartma

- İlk önce çıkan binary sayının 1 tamamlayıcısı alınır.
- İkincil olarak ana sayı ile çıkan sayının tamamlayıcısı toplanır.
- Son olarak en soldaki rakam sağdakilerle toplanır.
- Eğer işlem sonucunda artan 1 yok ise sonuç negatif olup, gerçek sonuç eldeki değerin 1 tamamlayıcısıdır.


1101 1001 1001
1010 0101 1101
1101 1001 1001
+ 0101 1010 0010
1
0011 0011 1011
+
1
0011 0011 1100

2'ye Tamamlama Yöntemiyle Çıkartma

- İlk önce çıkan binary sayının 2 tamamlayıcısı alınır.
- İkinci aşamada ana sayı ile çıkan sayının tamamlayıcısı toplanır.
- İşlem sonucu artan "1" olursa bu "1" sonucun pozitif olduğunu gösterir ve dikkate alınmaz.
- İşlem sonucunda artan "1" yoksa sonuç negatif demektir. Gerçek sonuç elde edilen sayının 2 tamamlayısı alınrak bulunur.


1101 1001 1001
1010 0101 1101
0101 1010 0010
+ 1
0101 1010 0011
(2 tamamlayıcılı hali)
1101 1001 1001
+
(Dikkate Alınmaz)
1 0011 0011 1100

-C- Bölme

Kurallar:

i) 0/0=0 ii) 0/1=0 iii)1/0=0 iv)1/1=1

-D- Çarpma

- Kurallar:

i) 0x0=0 ii) 1x0=0 iii)0/1=0 iv)1x1=1

Oktal Sayı Sistemi

- {0,1,2,3,4,5,6,7} rakamlarını kullanır.

Oktal-Desimal Çevrimi

Örn:

$(564)_8=(?)_{10}$

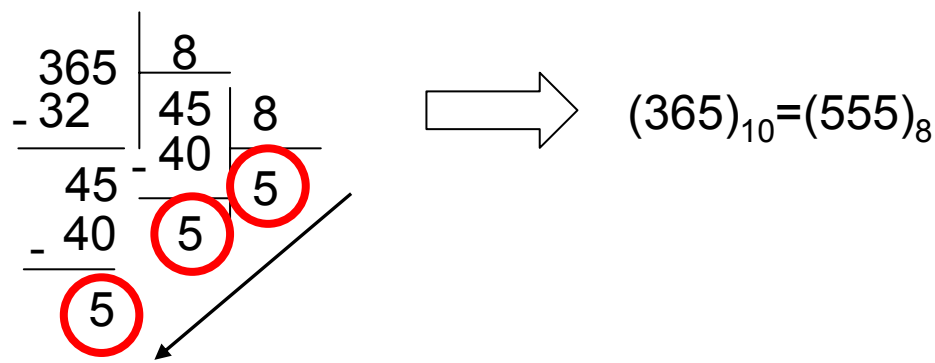
$5.8^2+6.8^1+4.8^0= (372)_{10}$

Desimal-Oktal Çevrimi

- Desimal sayıların oktal sayılara çevrimi "Bölme-8" yoluyla gerçekleştirilir. Çıkan sonuç tersten yazılır.

Örn:

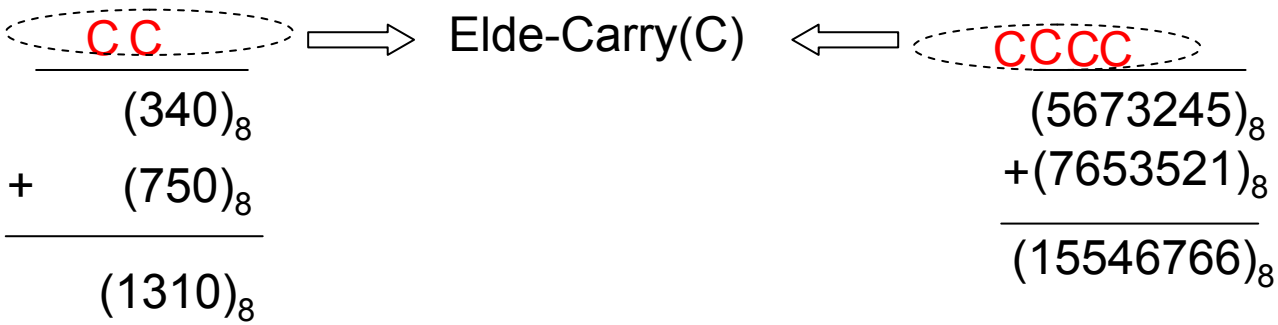
$(365)_{10}=(?)_8$



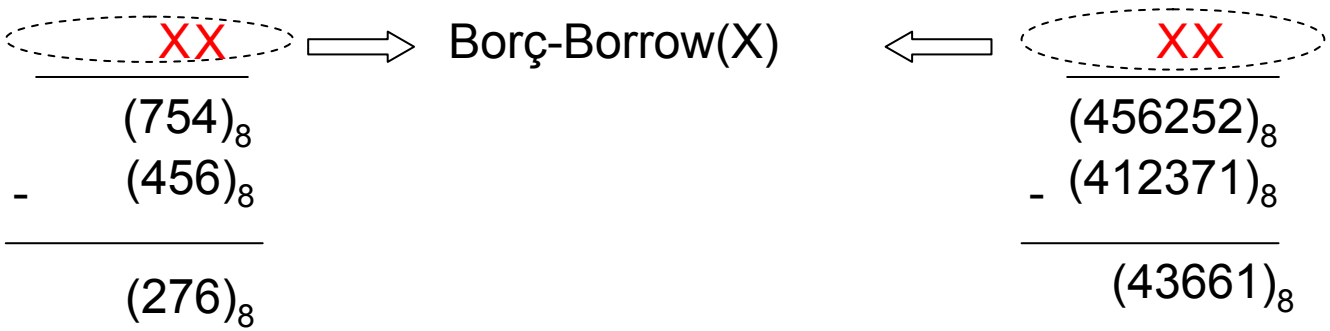
$(365)_{10}=(555)_8$

Oktal Sayı Sisteminde 4 İşlem

-A- Toplama



-B- Çıkartma



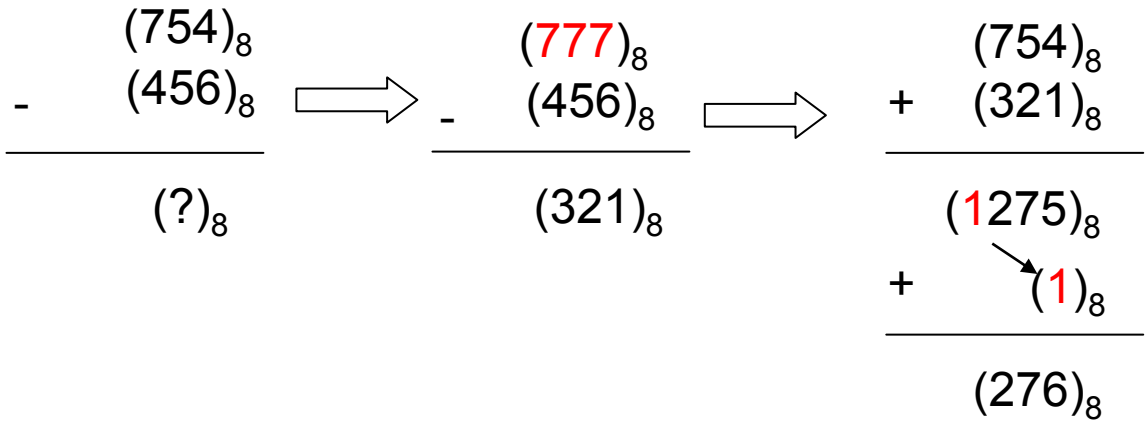
Tamamlayıcı Yardımıyla Çıkartma

- 7 ve 8 tamamlayıcısı olmak üzere iki adet tamamlayıcısı vardır. 7 tamamlayıcısı binary sistemdeki 1 tamamlayıcısına; 8 tamamlayıcısı ise 2 tamamlayıcısına benzer.
- 7 tamamlayıcısı bulunurken her dijitin 7 sayısından çıkarılması gerekir.
- 8 tamamlayıcısı = 7 tamamlayıcısı +1

7 tamamlayıcısı ile Çıkartma

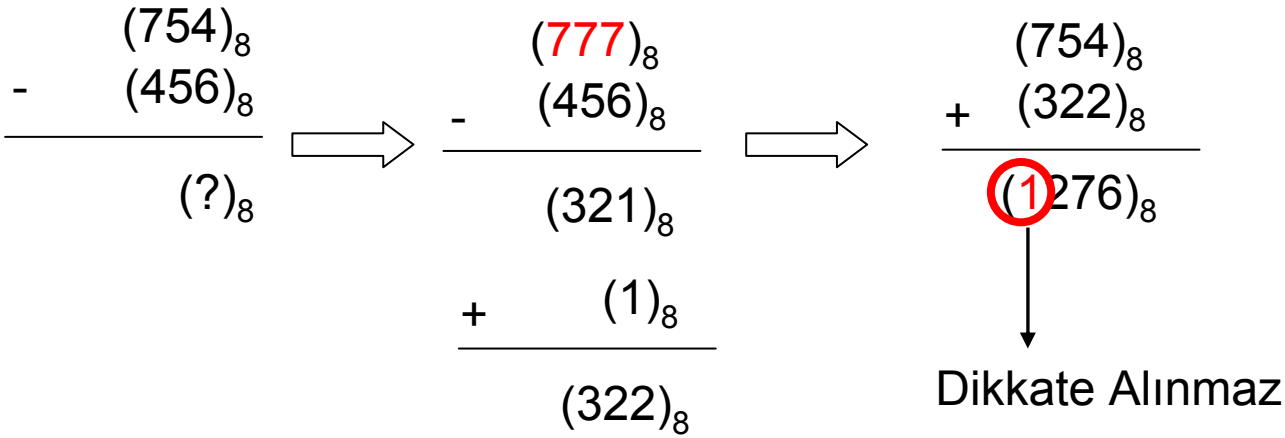
- İlk olarak oktal sayının 7 tamamlayıcısı bulunur.
- Ana sayı ile çıkan sayının 7 tamamlayıcısı toplanır.
- Eğer belirli basamak sayısını taşan “1” varsa sonuç pozitif olup, bu “1” sağdakilerle toplanılarak gerçek sonuç bulunur.
- Taşan “1” yoksa sonuç negatif olup, gerçek sonuç sayının 7 tamamlayıcısıdır.

Örn:



8 tamamlayıcısı ile Çıkartma

- Çıkan sayının 8 tamamlayıcısı bulunduktan sonra bu sayı ana sayı ile toplanır.
- Eğer belirli basamak sayısını taşan “1” var ise toplam sonucu pozitiftir ve bu “1” dikkate alınmaz
- Taşan “1” yok ise sonuç negatiftir ve gerçek sonuç sayının 8 tamamlayıcısıdır.


(754)8
- (456)8
(?)8
(777)8
- (456)8
(321)8
+ (1)8
(322)8
(754)8
+ (322)8
(1276)8
Dikkate Alınmaz

-C- Bölme İşlemi

En uygun yol ilk önce oktal-desimal çevrimini yaptıktan sonra bölme işlemini desimal ortamda gerçekleştirmek ve sonucu oktal sisteme dönüştürmektir.

-D- Çarpma İşlemi

En uygun yol ilk önce oktal-desimal çevrimini yaptıktan sonra çarpma işlemini desimal ortamda gerçekleştirmek ve sonucu oktal sisteme dönüştürmektir.

Heksadesimal Sayı Sistemi

- Tabanı 16'dır.
- "0....15" arası sayıları kullanır.

Desimal	Hex Karşılık
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Hex-Desimal Çevrimi

Örn:

$(FA54)_{16} = (\ ?\ )_{10}$

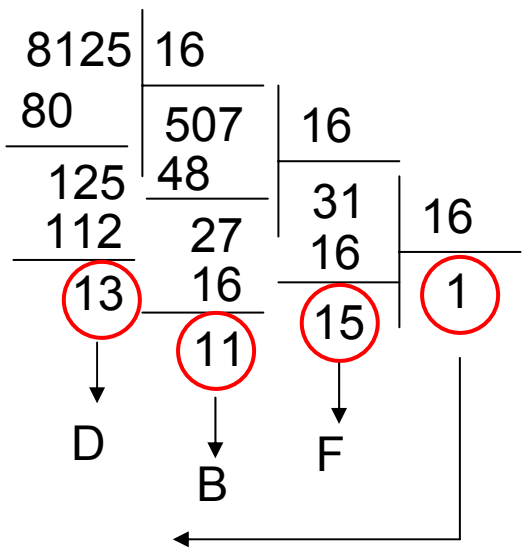
$=15.16^3+10.16^2+5.16^1+4.16^0$

$=(64084)_{10}$

Desimal-Hex Çevrimi

Örn:

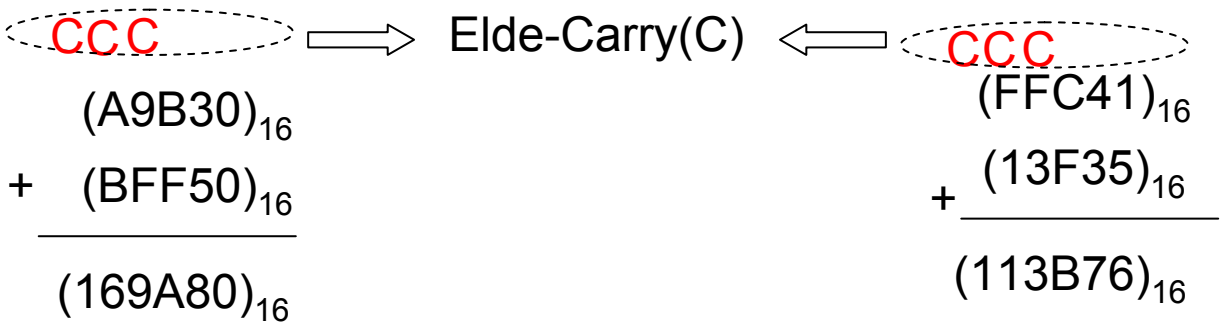
$(8125)_{10} = (?)_{16}$



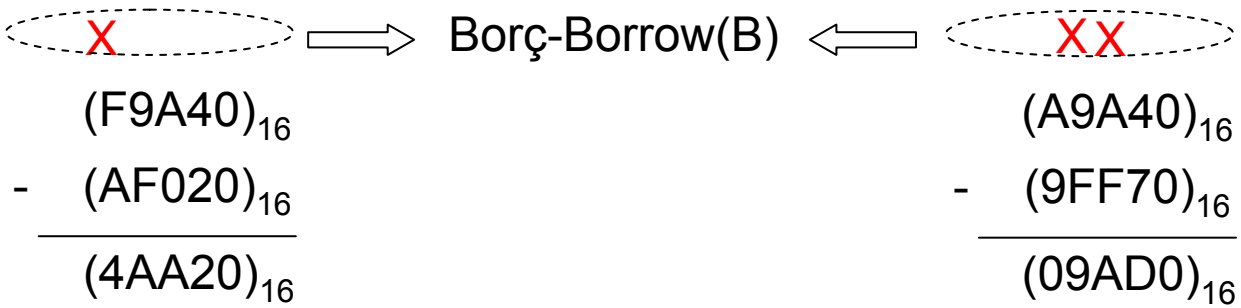
$(8125)_{10} = (1FBD)_{16}$

Heksadesimal Sayılarda 4 İşlem

-A- Toplama



-B- Çıkartma


X
Borç-Borrow(B)
XX
(F9A40)16
- (AF020)16
(4AA20)16
(A9A40)16
- (9FF70)16
(09AD0)16

Tamamlayıcı Yoluyla Çıkartma İşlemi

- 15 ve 16 tamamlayıcısı olmak üzere iki adet tamamlayıcı vardır. 15 tamamlayıcısı binarydeki 1 tamamlayıcısına ve oktaldaki 7 tamamlayıcısına benzer.
- 16 tamamlayıcsı binarydeki 2 tamamlayıcısına ve oktaldaki 8 tamamlayıcısına benzer.
- Sayının 1 tamamlayıcısı sayının F'den çıkarılmasıyla elde edilir.

Sayı Sistemleri Arasında Değişim

Binary-Oktal & Oktal-Binary

- Binary sayı oktala çevirirken, binary sayı sağdan başlayarak sola doğru 3'er 3'er gruplanır. Her grubun karşılığı bulunur.
- Oktal sayıyı binarye çevirirken bu işlemin tersi gerçekleştirilir.

- Örn:

$(11101001)_2=(?)_8$

$(011\ 101\ 001)_2 = (351)_8$

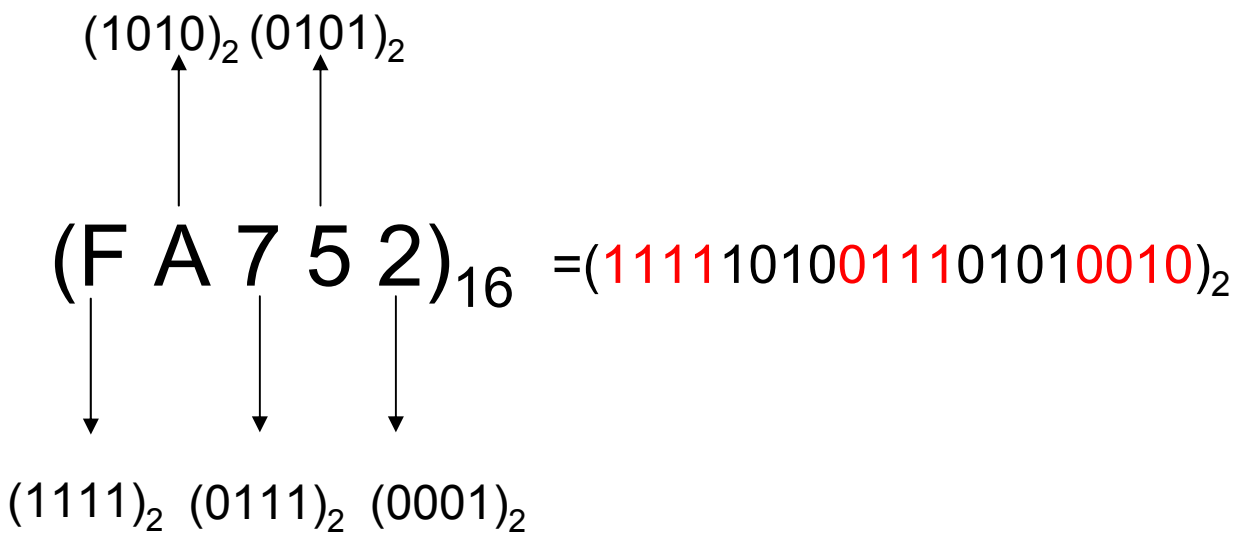
3 5 1

Binary-Hex & Hex-Bin

- Binary sayıyı heksadesimale çevirirken, binary sayı sağdan başlayarak sola doğru 4'er 4'er gruplandırılır. He grubun hex karşılığı bulunur.
- Heksadesimal sayıyı binarye çevirirken her heksadesimal sayı 4 bitlik sayı ile ifade edilir.

Örn:

$(FA752)_{16}=(?)_2$



Oktal-Hex & Hex-Oktal Çevirimi

- Bu çevrimi 2 şekilde yapmak mümkündür. Çevrim işlemi :
  - Binary sayılarla veya
  - Desimal sayılarla yapılabilir.

Boole Cebri

Boolean Cebri, Mantıksal Bağlaçlar, Lojik Kapılar ve Çalışma Mantıkları,

Doğruluk Tabloları, Boole Cebri Teoremleri, Lojik İfadelerin Sadeleştirilmeleri

Muhammet Baykara
mbaykara@firat.edu.tr

Boole Cebri

- Önermeler veya nesneler arasında ilişkileri betimler
- Simgesel matematiksel bir mantık sistemi
- 1854, George Boole, Mantık cebri
- 1938, Shannon, anahtar cebri
- Bilgisayarlarda kullanılan devrelerin tasarımı için gerekli temel

Boole Cebrinin Esasları

- Bir değişmez (sabit değişken), 1 ya da 0 şeklinde iki değer alabilir.
- İfadeler sabit değişkenin kendisi ya da tümleyenidir (değili/tersi).
- **Ve (AND)** bağlacı(., Λ), **Veya (OR)** bağlacı(+, V), **değil (NOT) (¯ , ')**

x	y	x.y xΛy	x+y x V y
Doğru	Doğru	Doğru	Doğru
Doğru	Yanlış	Yanlış	Doğru
Yanlış	Doğru	Yanlış	Doğru
Yanlış	Yanlış	Yanlış	Yanlış

x	y	x.y xΛy	x+y x V y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Boole Cebri Kuralları

- Değişme Kuralı:

  $X+Y= Y+X$ $\quad\quad$ $X.Y=Y.X$

- Birleşme Kuralı:

  $X+Y+Z=(X+Y)+Z=X+(Y+Z)$ $\quad\quad$ $X.Y.Z=(X.Y).Z=X.(Y.Z)$

- Dağılma Kuralı:

  $X.(Y+Z)=X.Y+X.Z$ $\quad$ $(X+Y).\ (X+Z) = X+Y.Z$

- Özdeşlik Kuralı:

  $X+X=X$

  $X.X=X$

Boole Cebri Kuralları

- VE Kuralı:

  $X.1=X \quad X.0=0$

- VEYA Kuralı

  $X+0=X \quad X+1=1$

- Tamamlayıcı Kuralı

  $x+\overline{x}=1$

  $x.\overline{x}=0$

- De Morgan Kuralı

$\overline{x.y}=\overline{x}+\overline{y}$

$\overline{x+y}=\overline{x}.\overline{y}$

- Tersin Tersi Kuralı

$\overline{\overline{X}}=X$

$\overline{X+Y}=\overline{X}+\overline{Y}$

$\overline{\overline{X.Y}}=X.Y$

Boole Cebri Kuralları

A, B ve C

$\overline{A} + B + \overline{C} = 0$?

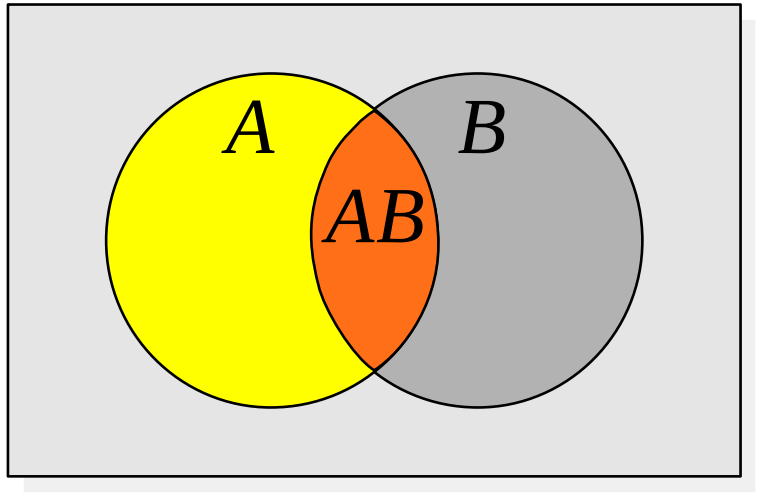
$A \cdot \overline{B} \cdot \overline{C} = 1$?

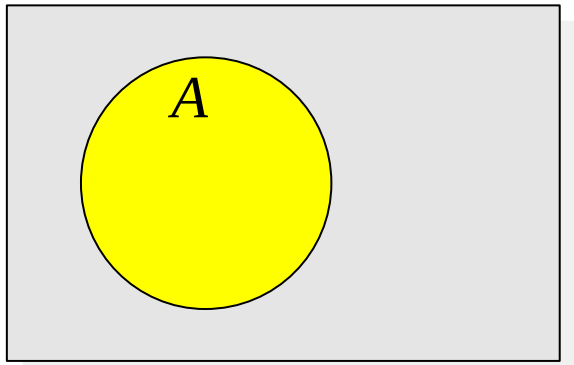
$A(A+\overline{B}) = ?$

$AB + A\overline{B} = ?$

Boole Cebri- Venn Şemaları Gösterimi

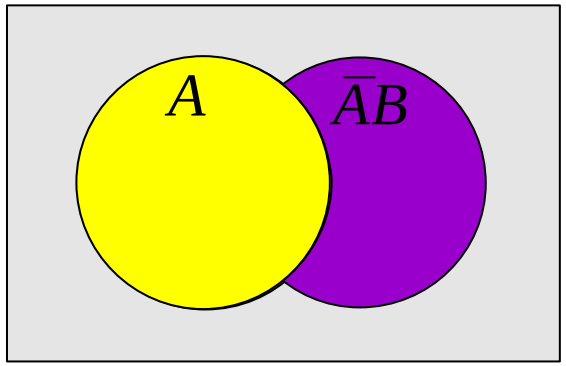
- $A + AB = A$



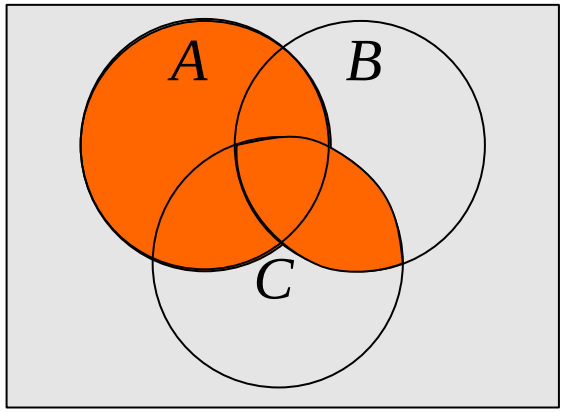


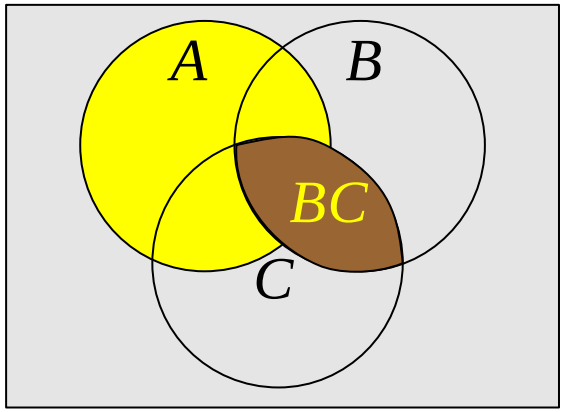
Boole Cebri- Venn Şemaları Gösterimi

- $A + \overline{A}B = A + B$



- $(A + B)(A + C) = A + BC$





Doğruluk Tabloları ve işlem sadeleştirme

x	y	x.y xΛy	x+y x V y	$\overline{x}$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

X.Y+X fonksiyonuna ait doğruluk tablosu?

Doğruluk Tabloları ve işlem sadeleştirme

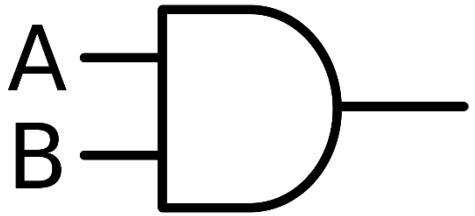
- $X.Y+\overline{X}.Y+\overline{Y}$ =?

- $X.Y.Z+X.Y.\overline{Z}+\overline{X}.Y$=?

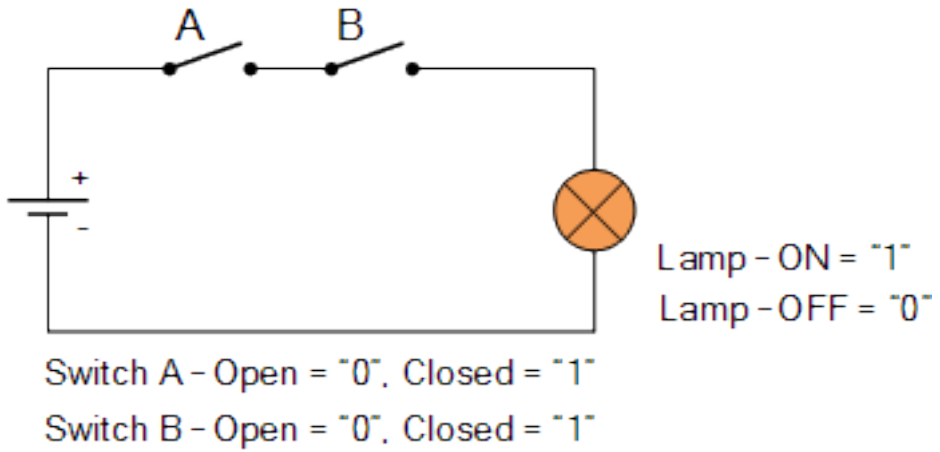
- $X.(\overline{X}+Y)$=?

Lojik Kapılar ve Doğruluk Tabloları

VE KAPISI (AND)

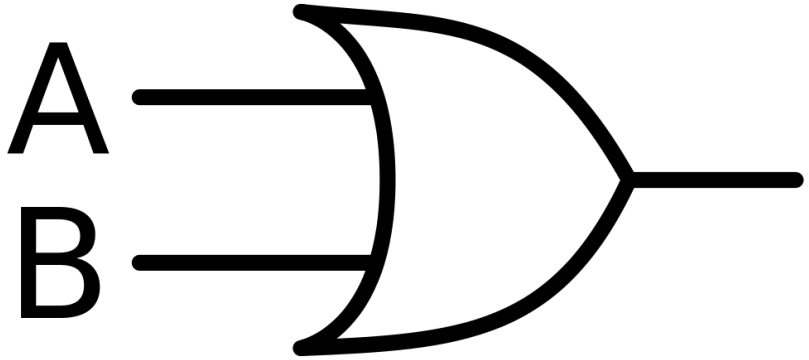


A	B	F=A.B
0	0	0
0	1	0
1	0	0
1	1	1

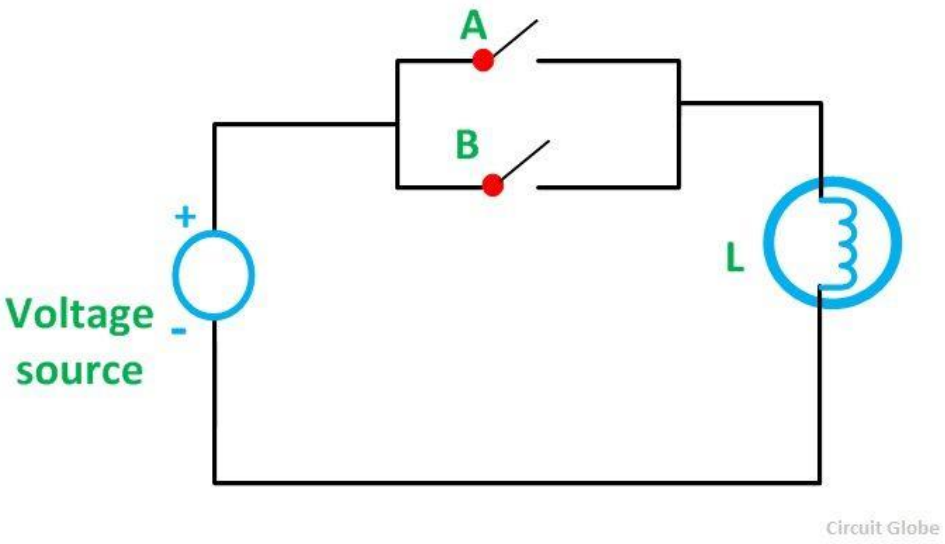


Lojik Kapılar ve Doğruluk Tabloları

- **VEYA KAPISI (OR)**



A	B	F=A+B
0	0	0
0	1	1
1	0	1
1	1	1

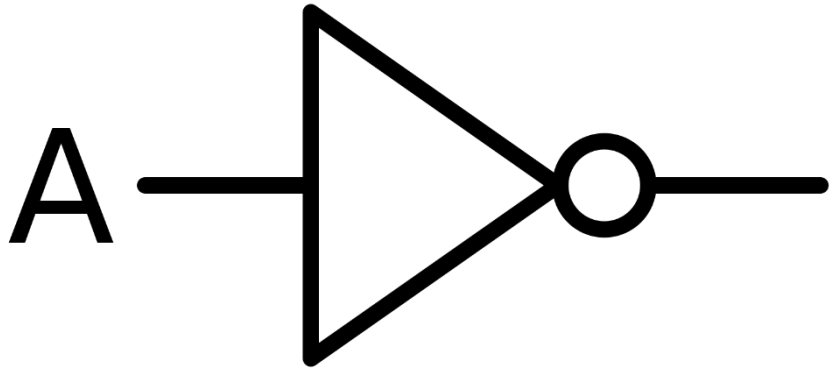


Lojik Kapılar ve Doğruluk Tabloları

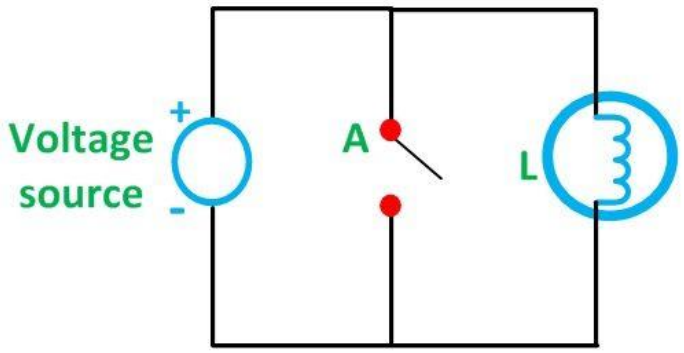
- Üç Girişli Veya Kapısı?

Lojik Kapılar ve Doğruluk Tabloları

- DEĞİL KAPISI (NOT)

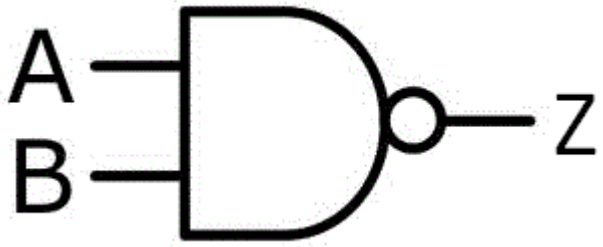


A	F
0	1
1	0

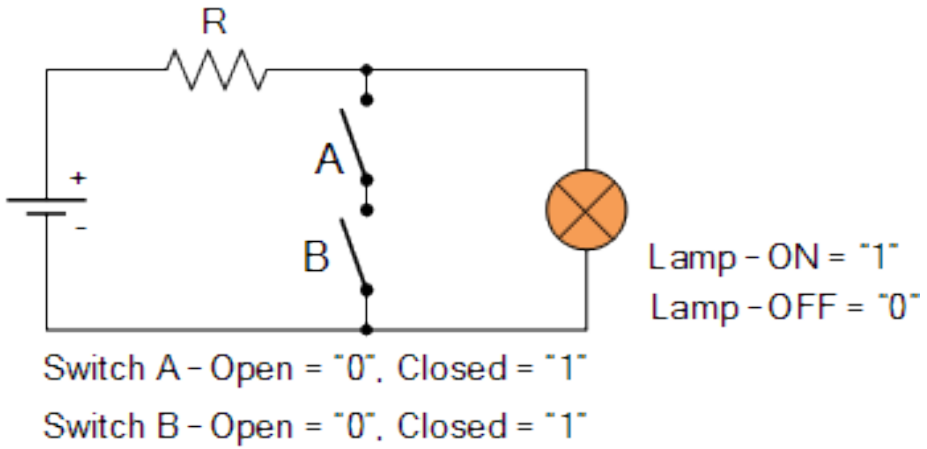


Lojik Kapılar ve Doğruluk Tabloları

- Ve DEĞİL KAPISI (NAND)

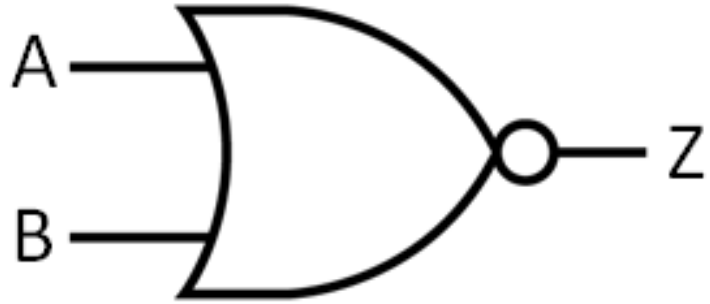


A	B	$Z=\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

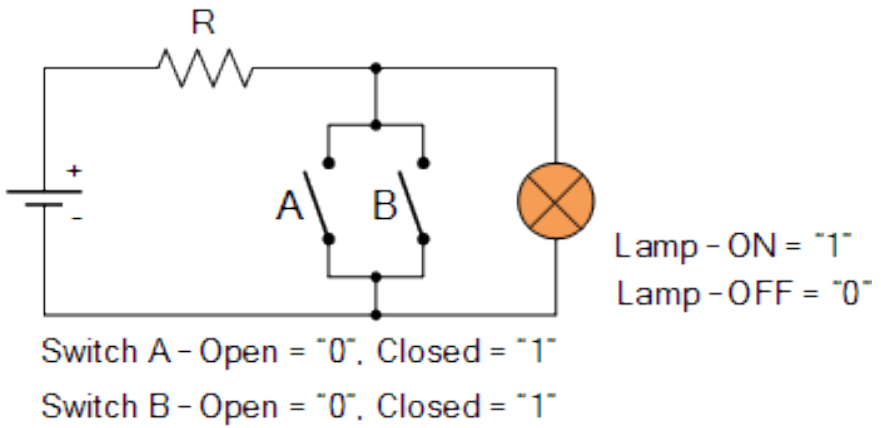


Lojik Kapılar ve Doğruluk Tabloları

- VEYA DEĞİL KAPISI (NOR)

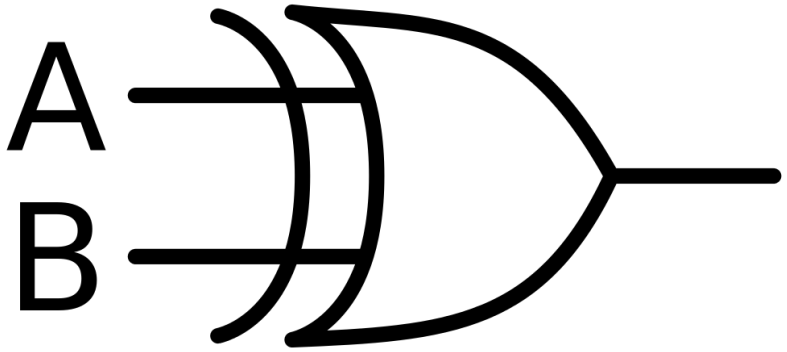


A	B	$Z=\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0



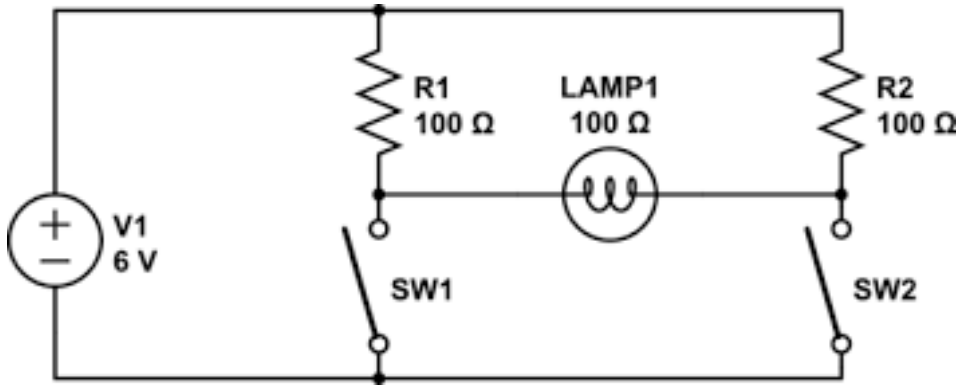
Lojik Kapılar ve Doğruluk Tabloları

- YA DA KAPISI (XOR)



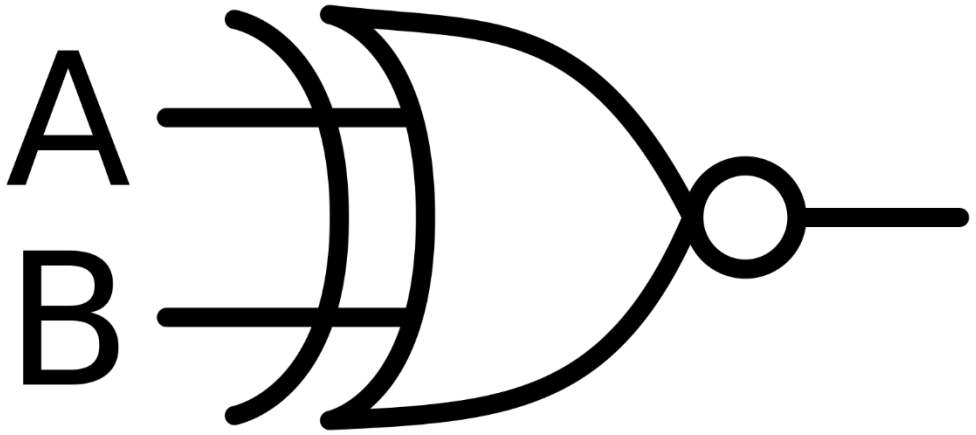
A	B	Z=A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

⊕ • $\overline{A}.B + A.\overline{B}$



Lojik Kapılar ve Doğruluk Tabloları

- YA DA DEĞİL KAPISI (XNOR)

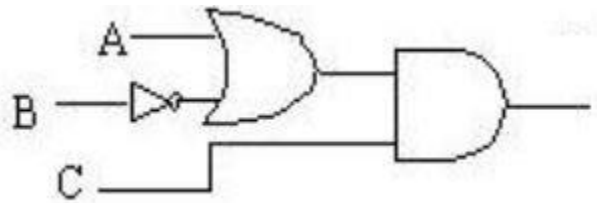


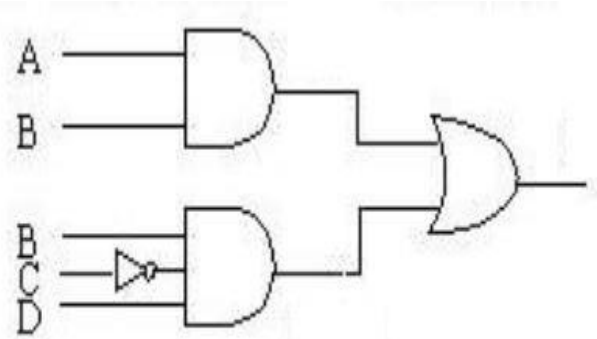
A	B	$Z=\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

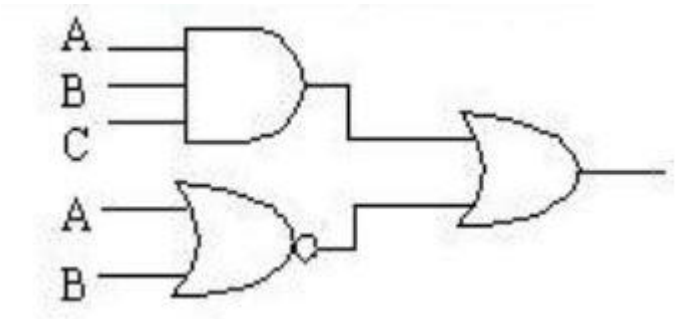
- $A.B+\overline{A}.\overline{B}$

Lojik diyagram örnekleri

- Lojik diyagramlardan mantıksal ifadenin elde edilmesi





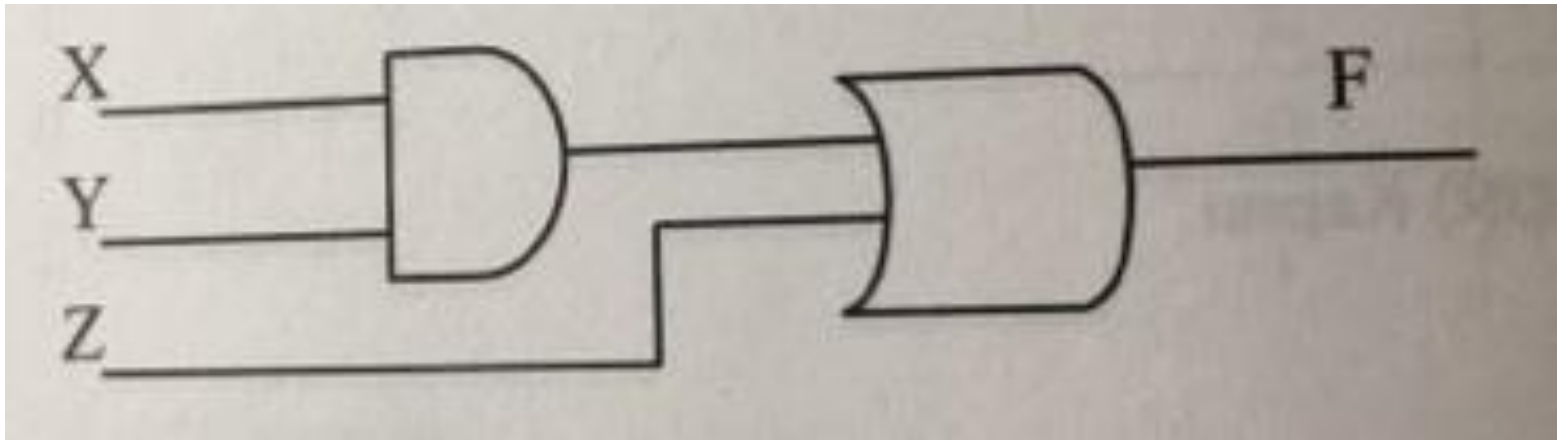


Lojik diyagram örnekleri

- Mantıksal eşitlikten Lojik Diyagramın elde edilmesi

Lojik Kapı Örnekleri-
Lojik Diyagramdan Matematiksel İfadenin Elde Edilmesi

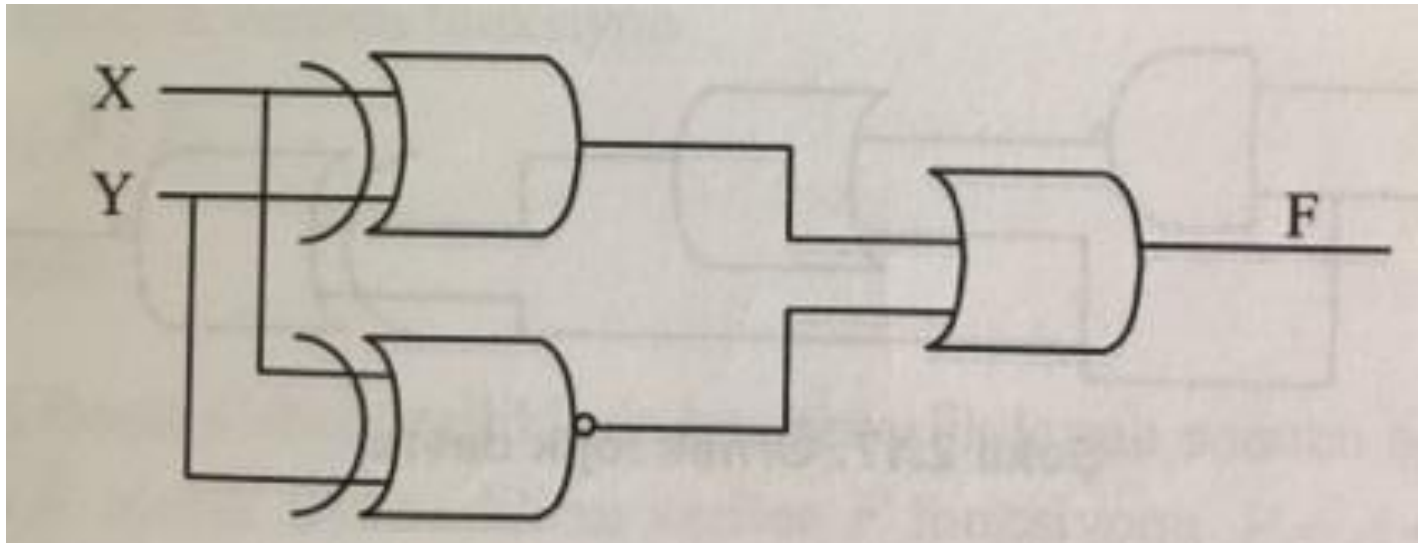
- Aşağıdaki devrenin matematiksel karşılığını bulunuz.



Şekil: VE VEYA Kapısı Örneği

Lojik Kapı Örnekleri-
Lojik Diyagramdan Matematiksel İfadenin Elde Edilmesi

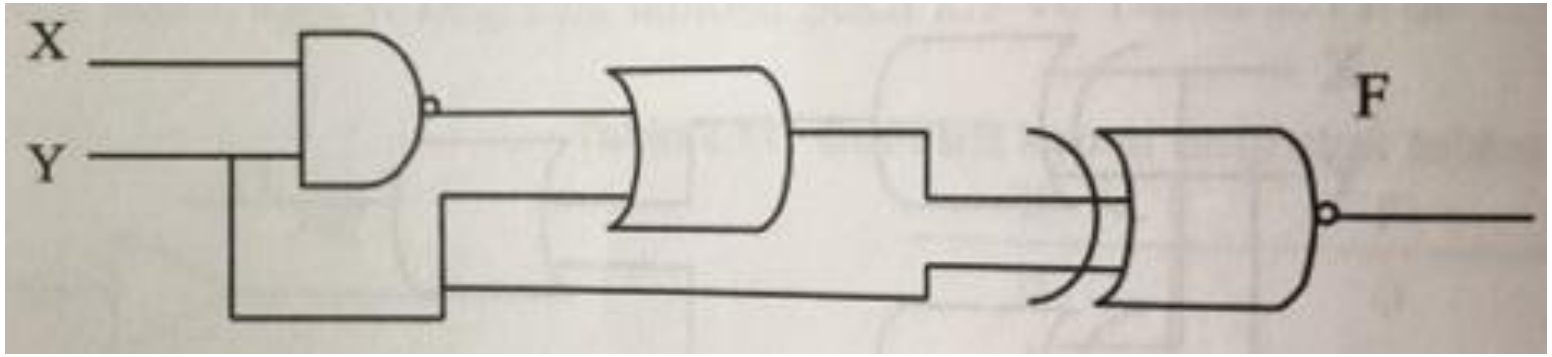
- Aşağıdaki devrenin matematiksel karşılığını bulunuz.



Şekil: YA DA VEYA İle Örnek Devre

Lojik Kapı Örnekleri-
Lojik Diyagramdan Matematiksel İfadenin Elde Edilmesi

- Aşağıdaki devrenin matematiksel karşılığını bulunuz.



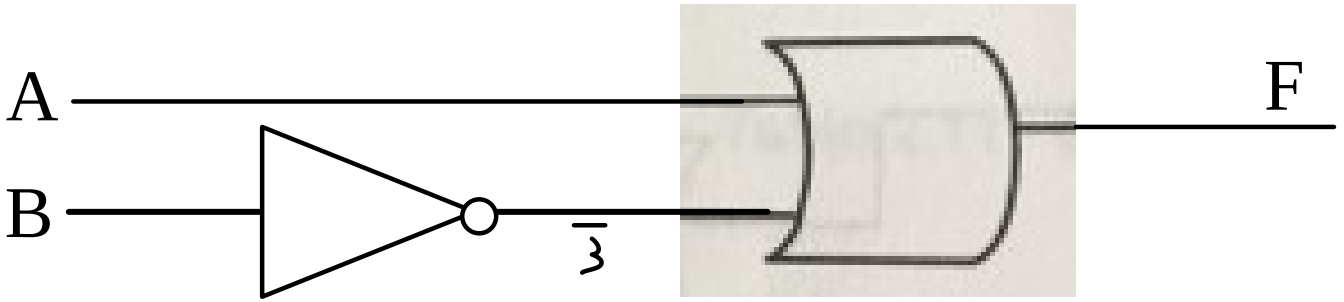
Şekil: Örnek Lojik Devre

Lojik Kapı Örnekleri-
Matematiksel Eşitlikten Lojik Diyagramın Elde Edilmesi

- Aşağıdaki matematiksel eşitlikten lojik diyagramı elde ediniz.

$$F=(\overline{\overline{A}B}).(A+\overline{B})$$

Lojik Kapı Örnekleri-
Matematiksel Eşitlikten Lojik Diyagramın Elde Edilmesi



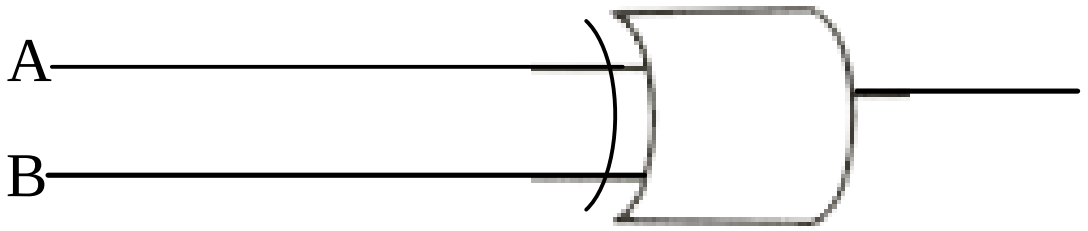
$F=A+\overline{B}$

Lojik Kapı Örnekleri-
Matematiksel Eşitlikten Lojik Diyagramın Elde Edilmesi

- Aşağıdaki matematiksel eşitlikten lojik diyagramı elde ediniz.

$$\overline{F=(\bar{A} + AB).(\bar{B}+AB)}$$

Lojik Kapı Örnekleri-
Matematiksel Eşitlikten Lojik Diyagramın Elde Edilmesi



$F=A\overline{B}+\overline{A}B=A\oplus B$

$F=A\overline{B}+\overline{A}B=A\oplus B$

Lojik İfadelerin Sadeleştirilmesi-Minumum Terimler (Minterm)

- Bitlik mintermlerin gösterilmesi

Değişkenler (x, y)		Minterm	Minterm Simgesi
0	0	$\overline{x} \cdot \overline{y}$	m_0
0	1	$\overline{x} \cdot y$	m_1
1	0	$x \cdot \overline{y}$	m_2
1	1	$x \cdot y$	m_3

Tabloda görüldüğü üzere minterm çarpım ile gösterilip değişkenlerin ‘0’ durumunda tümleyeni ($\overline{x}$, $\overline{y}$), ‘1’ durumunda kendisi (x , y) ile gösterilir. Örneğin; x $\overline{y}$, x’in ‘1’, y’nin ‘0’ olduğu durumu gösterir.

Lojik İfadelerin Sadeleştirilmesi-Minumum Terimler (Minterm)

- F (x, y)= $\bar{x}\,\bar{y} + x\,\bar{y} + xy$ ifadesini mintermlerle gösteriniz.

Lojik İfadelerin Sadeleştirilmesi-Minumum Terimler (Minterm)

- Bitlik minterm tablosu göz önüne alındığında verilen terimlere ilişkin ifade mintermler şeklinde,

- $F= m_0+ m_2+ m_3$veya

- $F (x, y) = \sum (0, 2, 3)$ gibi yazılabilir.

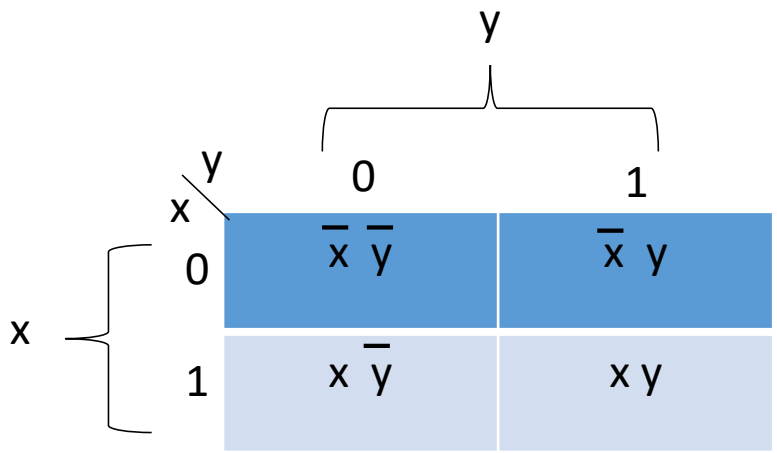
Lojik İfadelerin Sadeleştirilmesi-Maksimum Terimler (Maksterm)

- Bitlik maksterm tablosu göz önüne alındığında verilen terimlere ilişkin ifade makstermler şeklinde,

Değişkenler (x, y)		Maksterm	Maksterm Simgesi
0	0	$\overline{x} + \overline{y}$	M_0
0	1	$x + \overline{y}$	M_1
1	0	$\overline{x} + y$	M_2
1	1	$\overline{x} + \overline{y}$	M_3

Karnaugh Diyagramları-İki Değişkenli

m_0	m_1
m_2	m_3

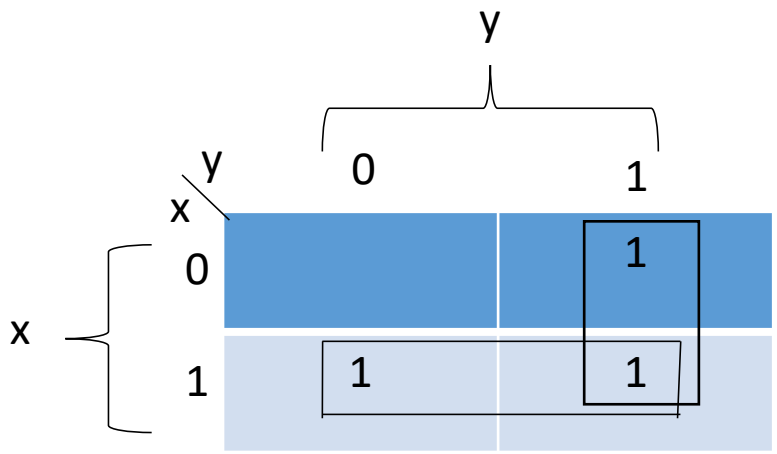


Karnaugh Diyagramları-İki Değişkenli

- F (x, y)= x + $\bar{x}$y fonksiyonunu **karnaugh** diyagramı ile sadeleştiriniz.

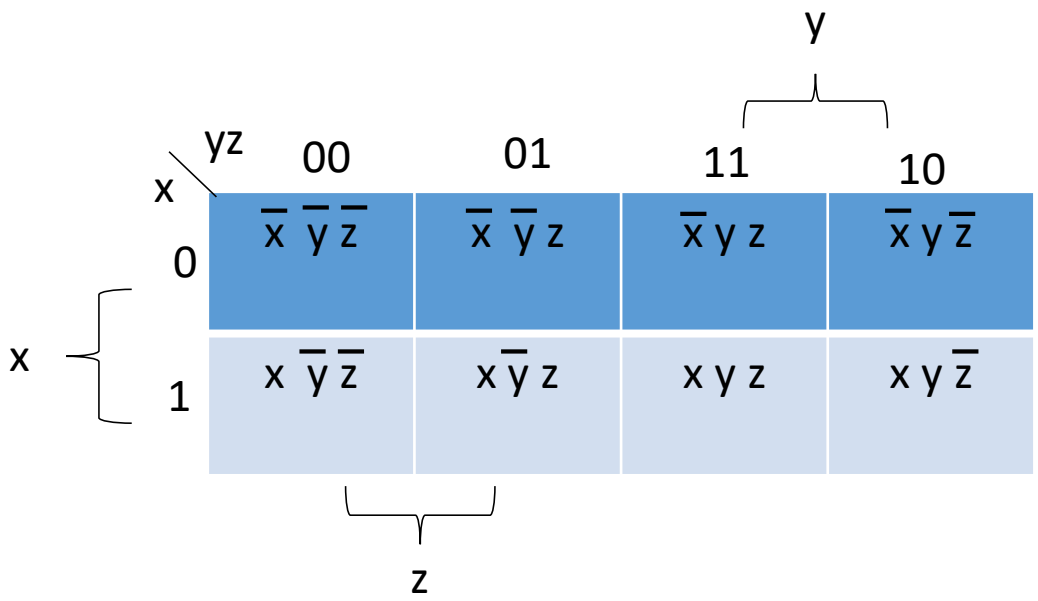
Karnaugh Diyagramları-İki Değişkenli

- F (x, y)= x + $\bar{x}$y fonksiyonu;



Karnaugh Diyagramları-Üç Değişkenli

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

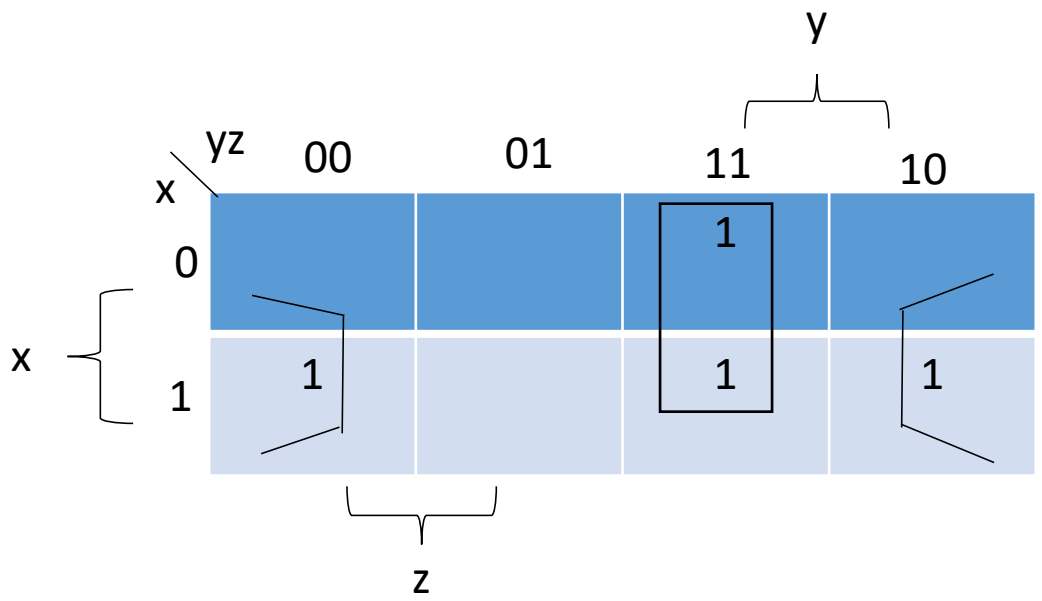


Karnaugh Diyagramları-Üç Değişkenli

F (x, y, z)= $\bar{x}$ y z + x y z + x $\bar{y}$ $\bar{z}$ + x y $\bar{z}$ fonksiyonunu **karnaugh** diyagramı yardımıyla sadeleştiriniz.

Karnaugh Diyagramları-Üç Değişkenli

F (x, y, z)= $\bar{x}$ y z + x y z + x $\bar{y}$ $\bar{z}$ + x y $\bar{z}$ fonksiyonu;

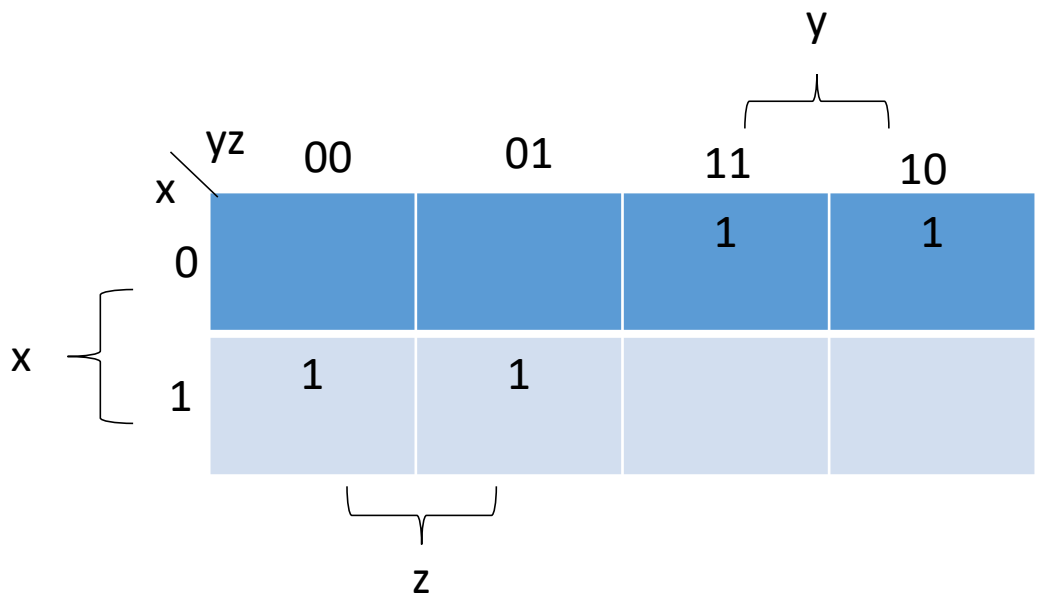


Karnaugh Diyagramları-Üç Değişkenli

F (x, y, z)= $x\,\bar{y}\,z + \bar{x}\,y\,\bar{z} + x\,\bar{y}\,\bar{z} + \bar{x}\,y\,z$ fonksiyonunu **karnaugh** diyagramı yardımıyla sadeleştiriniz.

Karnaugh Diyagramları-Üç Değişkenli

F (x, y, z)= x $\bar{y}$ z + $\bar{x}$ y $\bar{z}$ + x $\bar{y}$ $\bar{z}$ + $\bar{x}$ y z fonksiyonu;

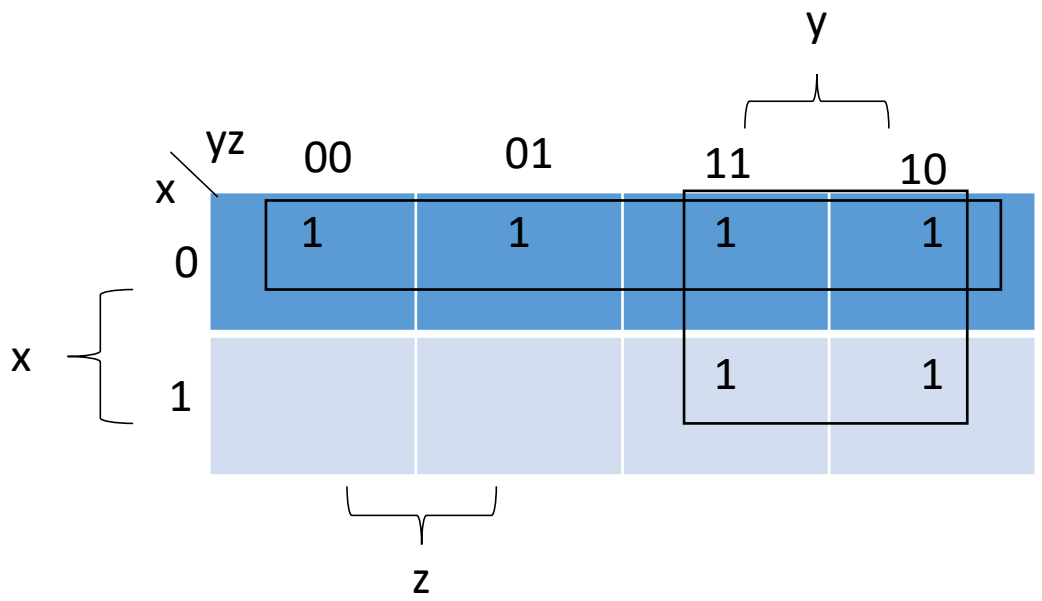


Karnaugh Diyagramları-Üç Değişkenli

F (x, y, z)= $\bar{x}$ + x y $\bar{z}$ + y z fonksiyonunu **karnaugh** diyagramı yardımıyla sadeleştiriniz.

Karnaugh Diyagramları-Üç Değişkenli

F (x, y, z)= $\bar{x}$ + x y $\bar{z}$ + y z fonksiyonu;



Algoritma ve Akış Şemaları

- Algoritma Tanımı,
- Algoritma Özellikleri,
- Algoritma Tasarımı,
- Akış Şemaları,
- Dallanma Simgeleri
- Döngü Simgeleri,
- Akış Şeması Tasarımı,
- Akış Şeması Özellikleri,
- N-S Şeması,
- W-O Diyagramları

Muhammet Baykara
mbaykara@firat.edu.tr

Algoritma

- **Algoritma** bir problemin çözümünde izlenecek yol anlamına gelir. Tüm programlama dillerinin temeli algoritmaya dayanmaktadır.
- İki farklı **algoritma oluşturma** şekli vardır. Birincisi **akış diyagramı** şeklinde oluşan algoritmalar, ikincisi ise **düz yazı** (**pseudo kod**) ile oluşturulan algoritmalardır. **Pseudo kod** dediğimiz kısım ile algoritmayı akış diyagramı yerine basit kelimeler ile açıklayıcı bir şekilde sıralamaya dayanır. Kısacası yapılacak adımları tek tek yazmak olarak tanımlanabilir.



Algoritmada Olması Gereken Özellikler

- **Etkin ve Genel Olma:** Algoritma etkin olmalı ve gereksiz tekrarlar bulunmamalıdır. Gerektiğinde diğer algoritmalar içerisinde kullanılabilmelidir. Her koşulda ve giriş değerinde doğru sonuca ulaşmalıdır.
- **Sonlu Olma:** Her türlü olasılık için algoritma sonlu adımda bitmelidir. Algoritma sonsuz döngüye girmemelidir.
- **Yanılmazlık:** Algoritma tekrar yürütüldüğünde aynı giriş değerleri için aynı sonuç elde edilmelidir.



Algoritmada Olması Gereken Özellikler

- **Giriş Çıkış Tanımlı Olma:** Algoritmalarda giriş ve çıkış bilgileri olmalıdır. Dışarıdan gelen verilere giriş bilgisi denir. Bu veriler algoritmada işlenir ve çıkış bilgisini oluşturur. Çıkış bilgisi her algoritmada mutlaka vardır. Algoritmaların temel amacı giriş bilgisini işleyerek çıkış bilgisi oluşturmaktır.
- **Başarım:** Amaç donanım gereksinimi (bellek kullanımı gibi), çalışma süresi gibi performans kriterlerini dikkate alarak yüksek başarımlı programlar yazmak olmalıdır.



Örnek Bir Algoritma (Doğrusal Akış Şemaları)

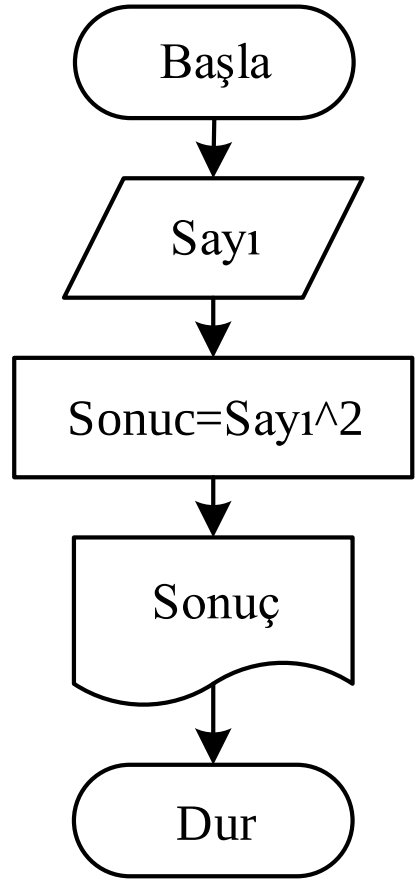
- En basit bir algoritma günlük doğal dil kullanılarak aşağıdaki gibi olabilir.

Algoritma 1:

1- Klavyeden girilen bir sayıyı oku.

2- Sayı üzerinde istenen işlemi yap; örneğin girilen sayının karesini hesapla.

3- Sonucu ekrana yaz.

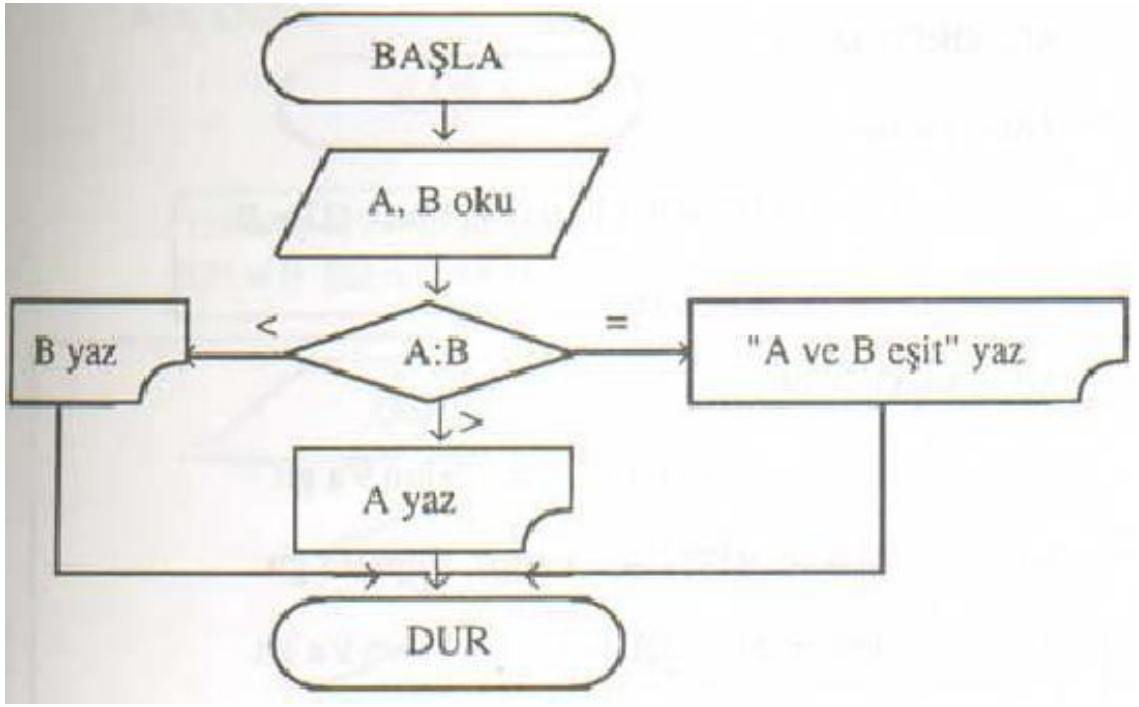


Örnek Bir Algoritma (MANTIKSAL AKIŞ ŞEMALARI)

- Geniş ölçüde mantıksal kararları içeren akış şemalarıdır. Hesap düzenleri genellikle basittir.

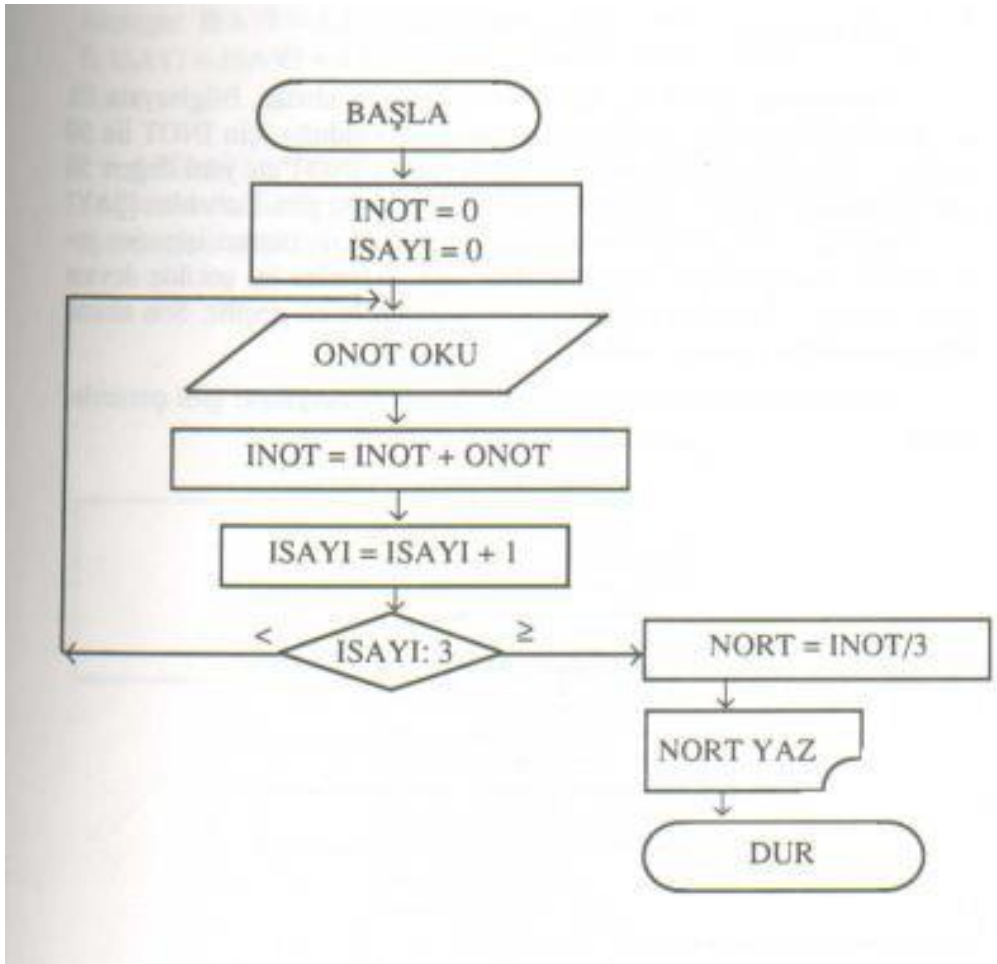
Algoritma 2:

- **Adım 1-**Başla
- **Adım 2-**A,B'yi oku
- **Adım 3-**A=B ise Adım 7'ye git
- **Adım 4-**A>B ise Adım 6'ya git
- **Adım 5-**B'yi yaz Adım 8'e git
- **Adım 6-**A'yı yaz Adım 8'e git
- **Adım 7-**"A ve B eşit' mesajını yaz
- **Adım 8-**DUR



Örnek Bir Algoritma (Yineli-Döngülü AKIŞ ŞEMALARI)

- Sorunun çözümü için, çözümde yer alan herhangi bir adım ya da aşamanın birden fazla kullanıldığı akış şemalarına denir.
- **Adım 1**-Başla
- **Adım 2**-INOT=0
- **Adım 3**-ISAYI=0
- **Adım 4**-ONOT oku
- **Adım 5**-INOT=INOT+ONOT
- **Adım 6**-ISAYI=ISAIY+1
- **Adım 7**-ISAYI<3ise Adım 4'e git
- **Adım 8**-NORT=INOT/3
- **Adım 9**-NORT YAZ
- **Adım 10**-DUR



Kaba (Pseudo) Kod ve Gerçek Kod

- **Kaba-kod,** bir algoritmanın yarı programlama dili kuralı, yarı konuşma diline dönük olarak ortaya koyulması/tanımlanmasıdır. Kaba-kod, çoğunlukla, bir veri yapısına dayandırılmadan algoritmayı genel olarak tasarlanır.
- **Gerçek kod** ise, algoritmanın herhangi bir programlama diliyle, belirli bir veri yapısı üzerinde gerçekleştirilmiş halidir.

```
connect (server_ip, server_port);
while(connected)
{
  /* if previous time slot ends
     start a new time slot */
  if (current_time_slot)
     continue;
  /* when receive input from
     player, send command to server */
  if (player_input( ))
     send_command( );
  /*when receive update from server
    update game status */
  if (recv_update( ))
     world_update( );
  /*predict for all entities*/
  prediction( );
  /*render graphic, play audio,...etc*/
  render_effect( );
}
```

```
while(!game_end)
{
  /*if any player time_out, end game*/
  if (!time_out)   continue;

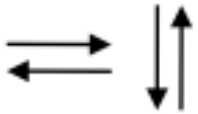
  while (recv_command( ))
  /*if players send a command, execute*/
  { execute_command( ); }
  /*AI determines NPC or item reaction*/
  perform_ai( );
  /*if player needs to be synchronized,
     sends update */
  for (n=0; n<player_number; n++)
  {
     if (update(n))
        send_update(n);
  }
}
```

client kodu | server kodu

```php
public static function fromXml($xml)
{
    // Extract parameters from the XML response
    $elements = @$xml->xpath('/response/data');
    if (empty($elements[0]))
    {
        $message = "Couldn't extract details from the XML";
        throw new InvalidArgumentException($message);
    }

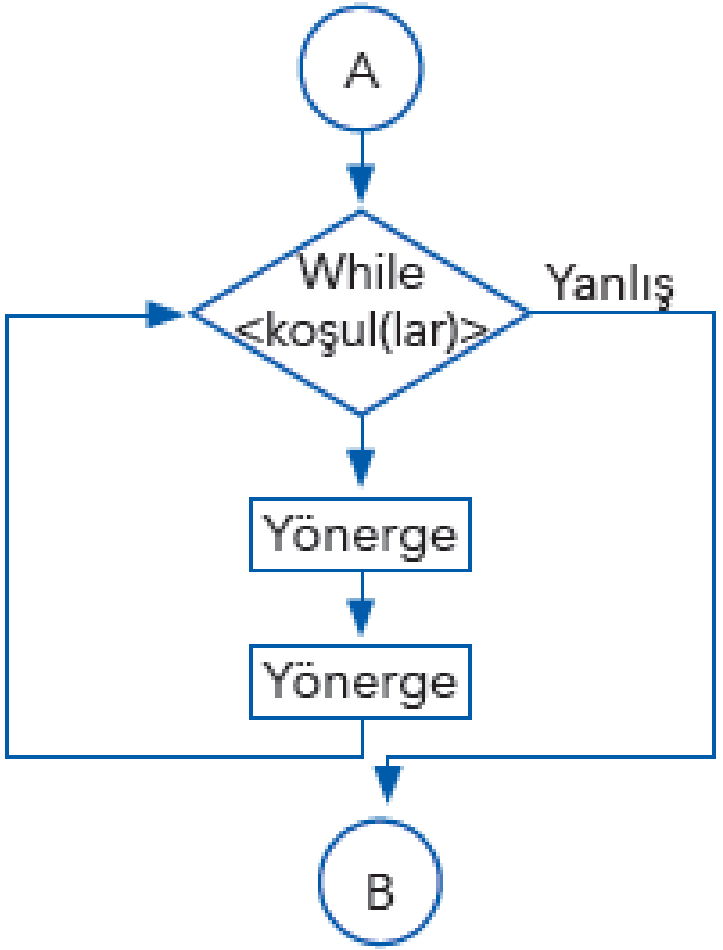
    $data = $elements[0];
    return new self(
        (float)   $data->amount,
        (string)  $data->currency,
        (string)  $data->id,
        (integer) $data->status,
        (string)  $data->status_msg
    );
}
```

Akış Şeması Simgeleri

Simge	İşlev	Simge	İşlev
	Başla/Bitir		Döngü
	Genel Girdi/Çıktı		Manyetik Disk
	Genel İşlem		Yordam Çağırma
	Denetim (Karar)		Akış Yönü
	Yazıcı Çıktısı		Bağlaç
	Görüntü Çıktısı		Sayfa Bağlacı
	Ele ile Girdi (Klavye)		

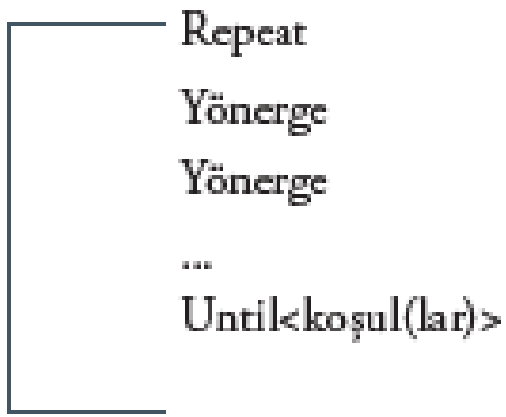
Döngüler

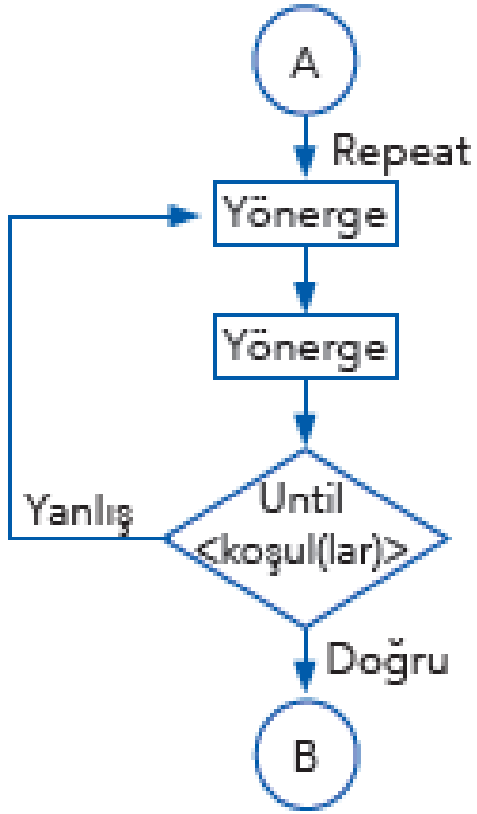
- Belirli bir iş bir çok kez tekrarlanacaksa, programda bu iş bir kez yazılır ve döngü deyimleriyle istenildiği kadar tekrar tekrar çalıştırılabilir.
- Bir döngüde, arka arkaya tekrarlanan deyimler döngü blokunu oluşturur. Bu deyimler birden çoksa { } bloku içine alınır. Bir döngü blokunda çok sayıda deyim olabileceği gibi, iç-içe döngüler de olabilir.
- Program akışının döngü blokunu bir kez icra etmesine döngünün bir adımı (bir tur) diyeceğiz.



Döngü Çeşitleri- Repeat-Until Döngüsü

- Repeat Until döngüsü belirteceğiniz bir şart doğru oluncaya kadar Repeat ile Until arasına yazacağınız bütün komutları tekrar tekrar başa giderek çalıştırır. Ta ki until ile belirttiğiniz şart doğru oluncaya kadar. Genel kullanımı:
- Repeat
- İşlem 1
- İşlem 2
- İşlem 3
- ...
- ...
- Until (sart);

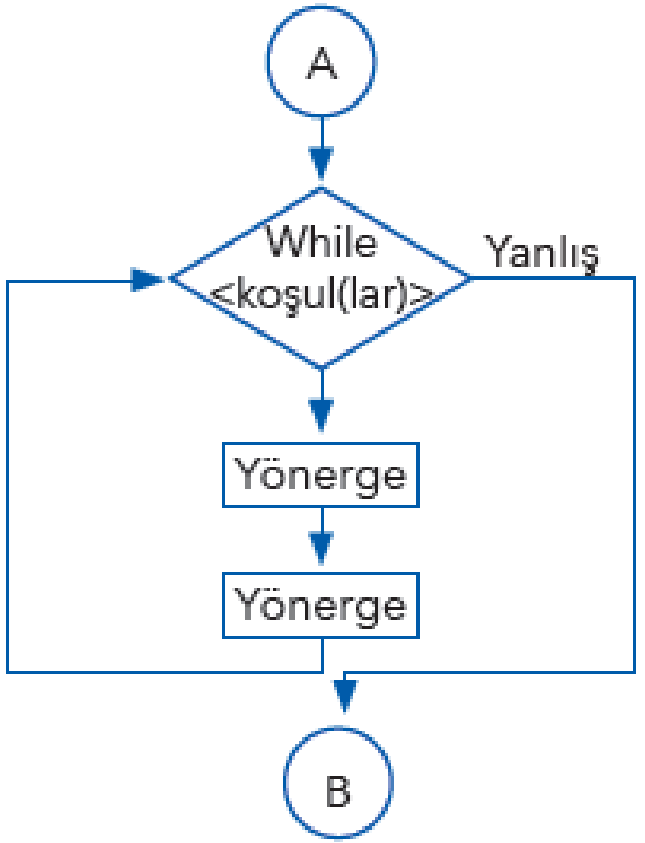




Döngü Çeşitleri- While Döngüsü

- Döngünün başlangıcında, döngü içerisinde yer alan yönergelerin işlenip işlenmeyeceğine ilişkin karar vermek için koşul kontrol edilir. Eğer koşul yanlış olursa döngü içerisindeki hiçbir yönerge işleme alınmaz. Eğer koşul doğru ise döngü içerisindeki tüm yönergeler çalıştırılır ve tekrar döngünün başına dönülür. Döngü, koşul durumu yanlış olana kadar devam eder.

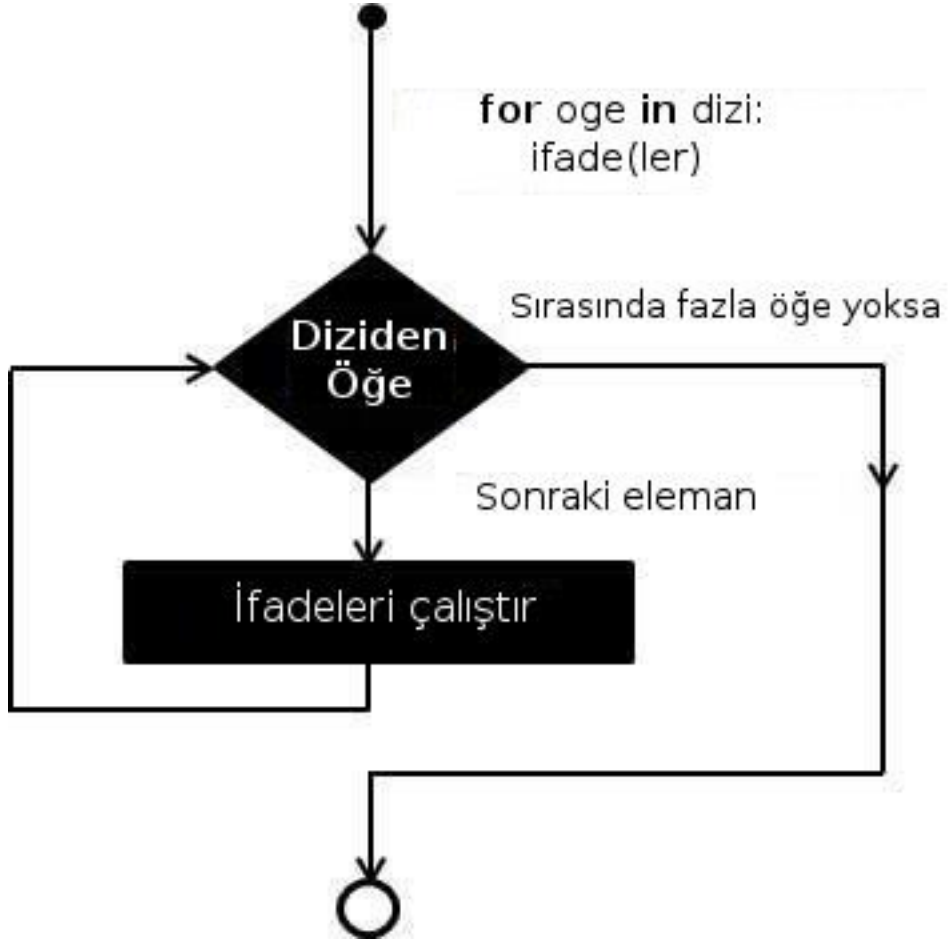
```
While<koşul(lar)>
Yönerge
Yönerge
……
WhileEnd
```



Döngü Çeşitleri- For Döngüsü

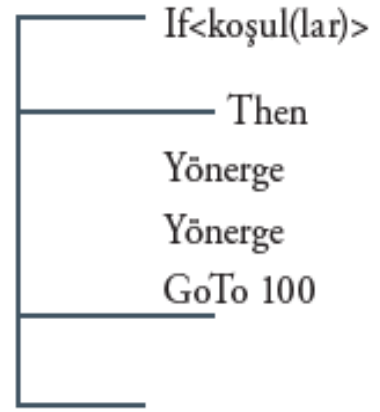
- *for deyimi* ve for deyimi kullanılarak oluşturulacak döngü yapısı, işlemlerin tekrar sayısının önceden belli olduğu durumlarda kullanılır.

```
for(ifade1;ifade2;ifade3)
        { Deyim1;
          Deyim2;
            ...
          Deyim_n;
        }
        Deyim_x;
```

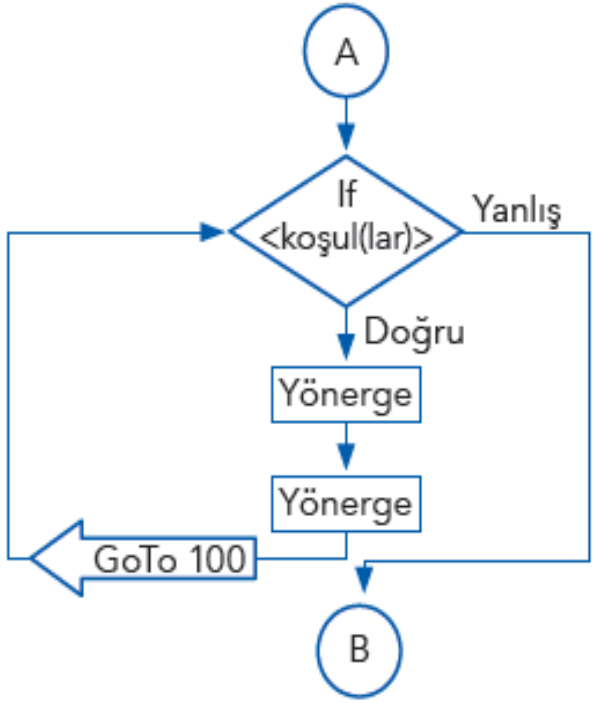


Döngü Bloklarına Döngü Dışından Gidilmesi

- Go To bir metod içerisinde bir işlemi yaptıktan sonra özellikle başka bir metoda gitmesi istenirs ya da bir switch case içerisinde tek değere göre birden fazla case gezdirmek gerekirse kullanılır.

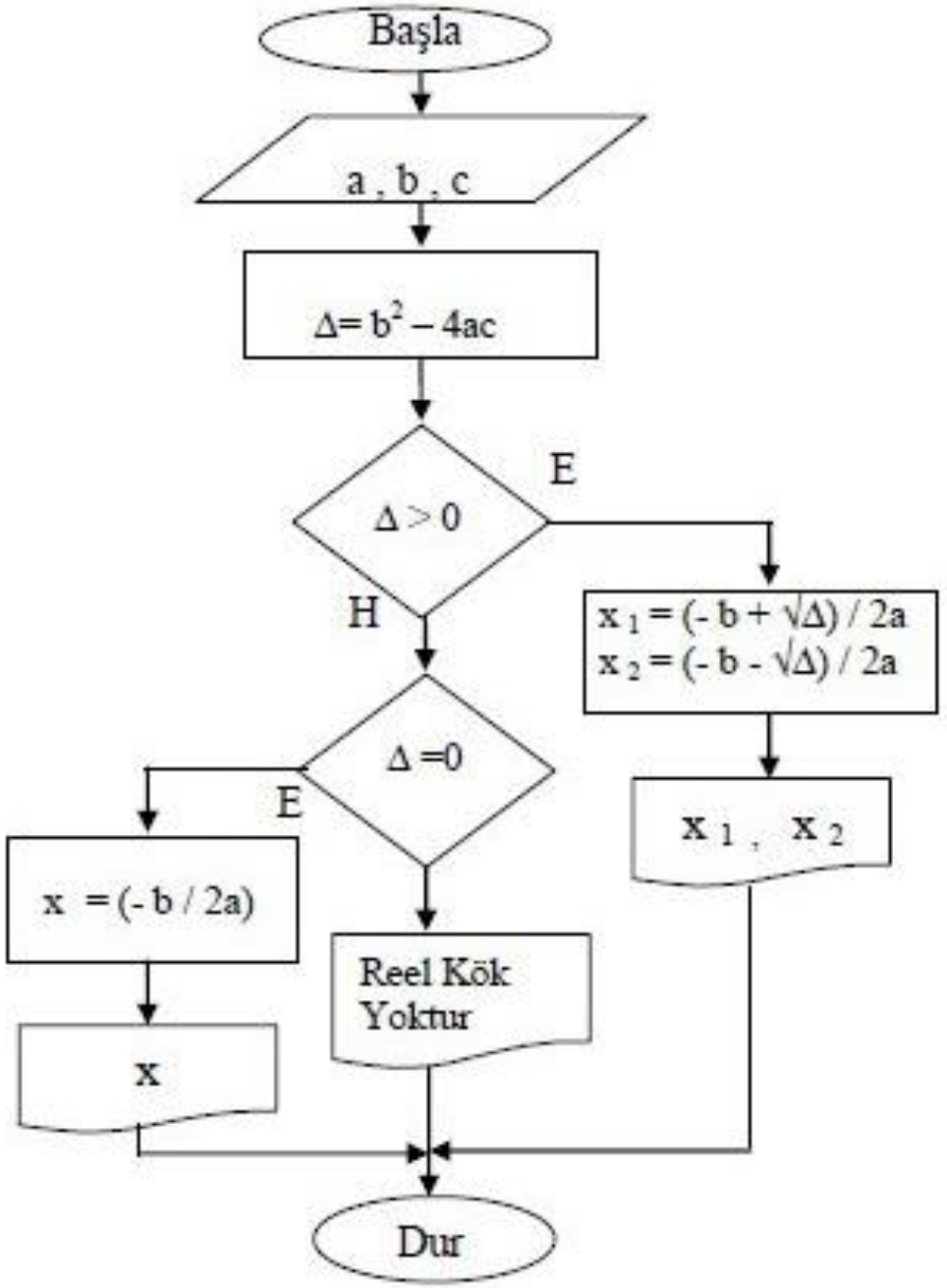



```
Etiketimiz:
int i = 0;
if (i == 0)
{
   MessageBox.Show("Cevabı Bildiniz");
   Close();
}
else
   goto Etiketimiz;
```



Akış Şeması Örneği

- 2. dereceden denklemin kökünü bulan akış şeması;



Algoritma ve İlişki Tanımlanması İçin Çeşitli Yöntemler

- Metinsel Tanımlama
- Akış Şeması
- N-S (Nassi-Schnederman) Şemaları
- W-O (Warnier-Orr) Şemaları
- --
- Bachman Notasyonu
- Crow’s Foot Spec Dili
- IDEF1XERD Spec Dili
- UML Yazılım Modelleme Dili
- SysML Sistem Modelleme

Sorularınız



Algoritma nedir?

Algoritma, belirli bir problemi çözmek veya belirli bir görevi gerçekleştirmek için adım adım yönergeler içeren bir bilgisayar programlama terimidir. Genel olarak, bir algoritma, giriş verilerini alır, bu veriler üzerinde belirli adımları gerçekleştirir ve bir çıktı üretir.

Algoritmaların temel amacı, belirli bir problemin çözümü için etkili, düzenli ve adım adım bir yönergeler dizisi sağlamaktır. İşte algoritmanın temel özellikleri:

Giriş (Input): Bir algoritma, belirli bir problemi çözmek veya bir görevi gerçekleştirmek için giriş verilerini alır. Bu veriler, algoritmanın çalışması için gereken bilgileri içerir.

Çıktı (Output): Algoritma, belirli bir işlemi tamamladıktan sonra bir çıktı üretir. Bu çıktı, problemi çözmek veya belirli bir görevi yerine getirmek için aranan sonuç olabilir.

Belirli Adımlar (Steps): Algoritma, belirli adımları takip ederek işlemi gerçekleştirir. Bu adımlar, problemin belirli bir sırayla çözülmesini veya görevin belirli bir düzen içinde tamamlanmasını sağlar.

Sonluluk (Finiteness): Algoritmanın bir süreç içinde belirli bir süre içinde sonlanması gerekir. Sonsuz döngülere veya belirsiz durumlara düşmemelidir.

Etkililik (Effectiveness): Algoritma, belirli bir problemin çözümü için etkili olmalıdır. Bu, verilen bir görevi mümkün olan en kısa sürede ve en az kaynak kullanarak tamamlama yeteneği anlamına gelir.

Algoritmalar, bilgisayar bilimlerinde, matematikte, mühendislikte ve birçok başka alanda geniş bir uygulama alanına sahiptir. İyi tasarlanmış algoritmalar, verimlilik, performans ve çözüm doğruluğu açısından önemli avantajlar sağlar.

Algoritmaları hangi yöntemlerle ifade edebiliriz? algoritmaların ifade edilme şekilleri nelerdir?

Algoritmalar, birçok farklı şekilde ifade edilebilir. İşte algoritmaların ifade edilme yöntemleri:

İngilizce veya Doğal Dil:

Algoritmalar, basit ve anlaşılır bir dil olan İngilizce veya başka bir doğal dil kullanılarak açıklanabilir. Bu, algoritmanın adım adım ne yapması gerektiğini anlatan metin tabanlı bir açıklama sağlar.

Akış Şemaları (Flowcharts):

Algoritmanın adımları, kararlar ve döngüler gibi yapısı, grafiksel bir şekil olan akış şemaları ile ifade edilebilir. Bu, algoritmanın yapısını görselleştirmenin etkili bir yoludur.

Pseudocode:

Pseudocode, bir programlama diline bağlı olmayan, insanların anlaması için yazılmış bir kod benzeri bir dilidir. Algoritmanın mantığını anlatmak için kullanılır ve genellikle resmi bir dil olmayan bir dilde yazılır.

Programlama Dilleriyle Kodlama:

Algoritmalar, programlama dilleri kullanılarak doğrudan kodlanabilir. Bu, algoritmanın gerçek bir bilgisayar programına dönüştürüldüğü anlamına gelir. Örneğin, C, Python, Java gibi programlama dilleri kullanılabilir.

Yapısal İşlemlerle Gösterim (Structured English):

İngilizce dilini kullanarak, programlama konseptlerini ve ifadelerini içeren yapısal bir ifade yöntemidir. Anlaşılabilir ve organize bir dil kullanarak algoritmayı ifade etmeye yöneliktir.

Matematiksel İfade:

Algoritmalar matematiksel ifadelerle de ifade edilebilir. Özellikle, sıralama, arama ve matematiksel problemler gibi belirli türdeki algoritmalar, matematiksel notasyonlar kullanılarak açıklanabilir.

Nesne Yönelimli Modellendirme (Object-Oriented Modeling):

Algoritmalar, nesne yönelimli programlama (OOP) yaklaşımıyla modellenebilir. Bu durumda, algoritmalar nesneler, metodlar ve ilişkilerle ifade edilebilir.

UML Diyagramları (Unified Modeling Language):

Algoritmalar, UML diyagramları aracılığıyla görselleştirilebilir. UML, yazılım mühendisliğinde kullanılan standart bir görsel modelleme dilidir.

Her bir ifade yöntemi, algoritmanın anlaşılmasını kolaylaştırmak ve uygulamayı geliştirmek için kullanılabilir. Hangi yöntemi seçeceğiniz, hedef kitlenizin anlama düzeyine, projenin gereksinimlerine ve sizin tercihlerinize bağlı olarak değişebilir.

Algoritmada Olması Gereken Özellikler Nelerdir?

İyi bir algoritmanın sahip olması gereken özellikler şunlar olabilir:

Belirginlik (Clarity): Algoritma, açık ve anlaşılır olmalıdır. Her adım, net bir şekilde tanımlanmış ve anlaşılır olmalıdır, böylece algoritmanın kullanıcıları veya başkaları tarafından anlaşılabilir.

Sonluluk (Finiteness): Algoritma, belirli bir adımdan sonra bir süre içinde sonlanmalıdır. Sonsuz döngülere veya belirsiz durumlara düşmemelidir.

Etkililik (Effectiveness): Algoritma, belirli bir görevi veya problemi etkili bir şekilde çözmelidir. Yani, giriş verileriyle başladığında, uygun adımları takip ederek beklenen çıktıyı üretebilmelidir.

Genellik (Generality): Algoritma, benzer problemleri veya görevleri çözmek için genelleştirilebilir olmalıdır. Sadece belirli bir duruma özgü değil, daha geniş bir uygulama alanına sahip olmalıdır.

Giriş ve Çıkış (Input ve Output): Algoritma, bir giriş almalı ve belirli bir çıktı üretmelidir. Bu, algoritmanın bir işlemi gerçekleştirebilmesi için gerekli olan verileri alabilmesini ve bir çıktı üretebilmesini sağlar.

Bağımsızlık (Independence): Algoritmanın çalışması, giriş verilerinden başka herhangi bir dış etken veya duruma bağlı olmamalıdır. Bağımsızlık, algoritmanın güvenilir ve tekrarlanabilir olmasını sağlar.

İzlenebilirlik (Traceability): Algoritma, her adımda ne yapılması gerektiğini açıkça belirttiği için izlenebilir olmalıdır. Bu, algoritmanın hatalarını bulmayı ve düzeltmeyi kolaylaştırır.

Verimlilik (Efficiency): İyi bir algoritma, kaynakları etkili bir şekilde kullanmalıdır. Zaman ve bellek kullanımını optimize etmek, algoritmanın performansını artırır.

Uygulanabilirlik (Feasibility): Algoritma, gerçek dünya koşullarında uygulanabilir olmalıdır. Pratikte kullanılabilecek ve belirli bir problemin çözümü için uygun olmalıdır.

Modülerlik (Modularity): Algoritma, modüler bir tasarıma sahip olmalıdır. Bu, algoritmayı küçük, bağımsız modüllere bölmeyi ve her bir modülü ayrı ayrı geliştirmeyi kolaylaştırır.

Bu özellikler, bir algoritmanın kaliteli, güvenilir ve uygulanabilir olmasını sağlamak için önemlidir. Algoritma tasarlarken, bu özelliklere dikkat ederek etkili ve sorunsuz çalışan algoritmalar oluşturmak önemlidir.

DOĞRUSAL ALGORİTMALAR:

Döngü ve kontrol yapıları içermeyen, sıralı işlemler içeren algoritmalardır.

Örnek: İki Sayının Toplamını Bulma

Başla
1. Birinci sayıyı A olarak al, örneğin A = 5
2. İkinci sayıyı B olarak al, örneğin B = 7
3. Toplam değişkenini A + B ile hesapla
4. Toplamı yazdır
Bitir

Örnek: İki Sayının Ortalamasını Bulma

Başla
1. Birinci sayıyı A olarak al, ~~örneğin A = 12~~
2. İkinci sayıyı B olarak al, ~~örneğin B = 8~~
3. İki sayının toplamını hesapla: C = A + B
4. İki sayının ortalamasını bul: Ortalama = C / 2
5. Ortalamayı yazdır
Bitir

Örnek: Dört İşlem Algoritması

Başla
1. Birinci sayıyı A olarak al, örneğin A = 15
2. İkinci sayıyı B olarak al, örneğin B = 7
3. Toplamı hesapla: Toplam = A + B
4. Farkı hesapla: Fark = A - B
5. Çarpımı hesapla: Carpim = A * B
6. Bölümü hesapla: Bolme = A / B
7. Toplamı yazdır
8. Farkı yazdır
9. Çarpımı yazdır
10. Bölümü yazdır
Bitir

MANTIKSAL ALGORİTMALAR:

Örnek: Bir Sayının Pozitif, Negatif veya Sıfır Olup Olmadığını Kontrol Etme

Başla
1. Bir sayıyı al, örneğin Sayi = 5
2. Eğer Sayi > 0 ise:
a. "Sayı pozitiftir" mesajını yazdır
3. Değilse, eğer Sayi < 0 ise:
a. "Sayı negatiftir" mesajını yazdır
4. Değilse:
a. "Sayı sıfırdır" mesajını yazdır
Bitir

Örnek: Bir Öğrencinin Not Durumunu Belirleme

Başla
1. Öğrencinin notunu al, örneğin Not = 75
2. Eğer Not >= 90 ise:
a. "Notunuz A" mesajını yazdır
3. Değilse, eğer Not >= 80 ise:
a. "Notunuz B" mesajını yazdır
4. Değilse, eğer Not >= 70 ise:
a. "Notunuz C" mesajını yazdır
5. Değilse, eğer Not >= 60 ise:
a. "Notunuz D" mesajını yazdır
6. Değilse:
a. "Notunuz F" mesajını yazdır
Bitir

DÖNGÜLÜ/ YİNELİ ALGORİTMALAR:

A1 : BAŞLA
A2 : i = 0
A3 : YAZ "Türkiye"
A4 : i = i + 1
A5 : EĞER i < 100 ise GİT A3
A6 : BİTİR

Başla
1. Bir sayı al, örneğin N = 5
2. Toplam değişkenini sıfırla
3. Bir döngü başlat: i = 1'den N'e kadar
a. Toplamı i ile güncelle: Toplam = Toplam + i
4. Döngü sonu
5. Toplamı yazdır
Bitir

ÖRNEKLER:

A1 : BAŞLA
A2 : sayac = 0
A3 : YAZ sayac
A4 : sayac = sayac + 2
A5 : EĞER sayac <= 100 ise GİT A3
A6 : BİTİR

```
A1 : BAŞLA
A2 : sayac = 1, toplam=0
A3 : toplam = toplam + sayac
A4 : sayac = sayac + 1
A5 : EĞER sayac <= 100 ise GİT A3
A6 : YAZ toplam
A7 : BİTİR
```

```
A1 : BAŞLA
A2 : sayac = 0, toplam=0
A3 : Oku sayi
A4 : toplam = toplam + sayi
A5 : sayac = sayac + 1
A6 : EĞER sayac < 10 ise GİT A3
A7 : YAZ toplam
A8 : BİTİR
```

```
A1 : BAŞLA
A2 : sayac = 1, s = 1
A3 : OKU sayi
A4 : s = s * sayac
A5 : sayac = sayac + 1
A6 : EĞER sayac <= sayi ise GİT A4
A7 : YAZ s
A8 : BİTİR
```

- ```
  x=3;
      if (x>5)
          then
                y=2*x
                z=y*y
          else
                y=x+4
                 z=y*y
  ```
  ```

YMT219
Veri Yapıları

Yrd.Doç.Dr. Erkan TANYILDIZI

Ders Kitapları ve Yardımcı Kaynaklar

- Veri Yapıları ve Algoritmalar
  - Dr. Rifat ÇÖLKESEN, Papatya yayıncılık



- Data Structures and Problem Solving Using Java
  - Mark Allen Weiss, Pearson International Edition

- Ahmet Yesevi Üniversitesi Ders Notları

- Ayrıca internet üzerinden çok sayıda kaynağa ulaşabilirsiniz.

Dersin Gereksinimleri

- Bu dersteki öğrencilerin Nesne tabanlı programlama dillerinden birisini(Java, C++, C#) veya yordamsal programlama dillerinden birisini(C, Pascal) bildiği varsayılmıştır.
- Bilinmesi gereken konular:
  - Temel veri türleri (int, float)
  - Kontrol yapısı (if else yapısı)
  - Döngüler
  - Fonksiyonlar(Methods)
  - Giriş çıkış işlemleri
  - Basit düzeyde diziler ve sınıflar

Ders İşleme Kuralları

- Derse devam zorunludur. En fazla 4 hafta devamsızlık yapılabilir.
- Ders başlangıç saatlerine özen gösteriniz. Derse geç gelen öğrenci ara verilinceye kadar bekleyecektir.
- Her ders iki imza alınacaktır.
- Ödevler zamanında teslim edilecektir. Verilen tarihten sonra getirilen ödevler kabul edilmeyecektir.
- Ders esnasında lütfen kendi aranızda (veya kendi kendinize) konuşmayın, fısıldaşmayın, mesajlaşmayın v.s
- Dersi anlatan ve dinleyen kişilere lütfen saygı gösterin.
- Anlatılan, anlatılmayan, merak ettiğiniz her konuda soru sor maktan çekinmeyin.
- Cep telefonu v.b kişisel taşınabilir iletişim cihazlarınızı ders süresince mutlaka kapalı tutunuz.

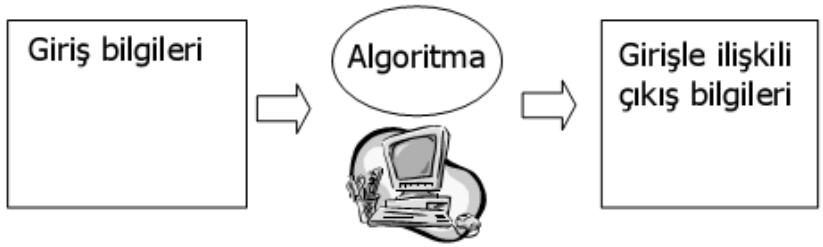
Veri Yapıları ve Modelleri

Bölüm 1

Algoritma

- **Algoritma**, bir problemin çözümünde izlenecek yol anlamına gelir. Algoritma, bir programlama dilinde (Java, C++, C# gibi) ifade edildiğinde **program** adını alır.

- Algoritma, belirli bir problemin sonucunu elde etmek için art arda uygulanacak adımları ve koşulları kesin olarak ortaya koyar. Herhangi bir giriş verisine karşılık, çıkış verisi elde edilmesi gereklidir. Bunun dışındaki durumlar algoritma değildir.



Algoritma

- Bilgisayar uygulamasında, bir yazılım geliştirirken birçok algoritmaya ihtiyaç duyulur. Örneğin,
  - arama algoritması
  - sıralama algoritması
  - matris veya vektörel işlem algoritması
  - graf algoritması
  - bir matematiksel modelin çözülmesi algoritması
- gibi birçok algoritma türü vardır ve uygulama geliştirirken, bunların biri veya birkaçı her zaman kullanılır.

Veri

- **Veri,** algoritmalar tarafından işlenen en temel elemanlardır (sayısal bilgiler, metinsel bilgiler, resimler, sesler ve girdi, çıktı olarak veya ara hesaplamalarda kullanılan diğer bilgiler…).

- Bir algoritmanın etkin, anlaşılır ve doğru olabilmesi için, algoritmanın işleyeceği verilerin düzenlenmesi gerekir.

Veri Yapısı ve Veri Modeli

- **Veri yapısı** (Data Structure) verinin veya bilginin bellekte tutulma şeklini veya düzenini gösterir.
  - Tüm programlama dillerinin, genel olarak, tamsayı, kesirli sayı, karakter ve sözcük saklanması için temel veri yapıları vardır. Bir program değişkeni bile basit bir veri yapısı olarak kabul edilebilir.
- **Veri modeli** (Data Model), verilerin birbirleriyle ilişkisel veya sırasal durumunu gösterir; problemin çözümü için kavramsal bir yaklaşım yöntemidir denilebilir.
  - Bilgisayar ortamında uygulanacak tüm matematik ve mühendislik problemleri bir veri modeline yaklaştırılarak veya yeni veri modelleri tanımlaması yapılarak çözülebilmektedir.

Veri Yapılarına Neden İhtiyaç Vardır?

- Bilgisayar yazılımları gün geçtikçe daha karmaşık bir hal almaktadır.
  - Örneğin 8 milyar sayfanın indekslenmesi (Google)
- Yazılımların programlanması ve yönetimi zorlaşmaktadır.
- Temiz kavramsal yapılar ve bu yapıları sunan çerçeve programları, daha etkin ve daha doğru program yazmayı sağlar.

Veri Yapılarına Neden İhtiyaç Vardır?

- İyi bir yazılım için gereksinimler:
  - Temiz bir tasarım
  - Kolay bakım ve yönetim
  - Güvenilir
  - Kolay kullanımlı
  - Hızlı algoritmalar
- Verimli Veri Yapıları
- Verimli Algoritmalar

Veri Yapılarına Neden İhtiyaç Vardır?

- **Örnek**
- Her biri satır başına ortalama 10 kelimeden ve yine ortalama 20 satırdan oluşan 3000 metin koleksiyonu olduğunu düşünelim.
  - →600,000 kelime
- Bu metinler içinde “dünya” kelimesi ile eşleşecek bütün kelimeleri bulmak isteyelim
- Doğru eşleştirme için yapılacak karşılaştırmanın 1 sn. sürdüğünü varsayalım.
- Ne yapılmalıdır?

Veri Yapılarına Neden İhtiyaç Vardır?

- **Örnek**
- **Çözüm. 1:**
- Sıralı eşleştirme: 1 sn. x 600,000 kelime= 166 saat
- **Çözüm. 2:**
- İkili Arama (Binary searching):
  - - kelimeler sıralanır
  - - sadece tek yarıda arama yapılır
  - toplam adım sayısı $\log_2 N = \log_2 600000$ yaklaşık 20 adım (çevrim) 20 sn.

- 20 saniye veya 166 saat!

Veri Yapılarına Neden İhtiyaç Vardır?

- **Örnek** :25 değerini 5 8 12 15 15 17 23 25 27 dizisinde arayalım. Kaç adımda sonuç bulunur?
  - 25 ? 15 15 17 23 25 27
  - 25 ? 23 23 25 27
  - 25 ? 25

Veri Yapılarının Sınıflandırılması

- Veri yapıları,
  - **Temel Veri Yapıları**
  - **Tanımlamalı (Bileşik) Veri Yapıları**
- Temel veri yapıları, daha çok programlama dilleri tarafından doğrudan değişken veya sabit bildirimi yapılarak kullanılır.
- Tanımlamalı veri yapıları, kendisinden önceki tanımlamalı veya temel veri yapıları üzerine kurulurlar; yani, önceden geçerli olan veri yapıları kullanılarak sonradan tanımlanırlar.

Veri Yapılarının Sınıflandırılması

- Programlama dilinin elverdiği ölçüde, hemen her tür veri yapısı tanımlanabilir. C Programlama dilinde yeni veri yapısı tanımlamak için struct, union gibi birkaç deyim vardır.

- Aşağıdaki bildirime göre **tam**, **kr** ve **kesirli** adlı değişkenler, C programlama dilinde birer temel veri yapısıdır; ancak, **toplam** adlı değişken ise, tanımlamalı veri yapısı şeklindedir. struct karmasik adlı veri yapısının 2 tane üyesi vardır; biri **gerçel**, diğeri **sanal** kısmı tutmak için kullanılır.

```
int tam ;
char kr ;
float kesirli;
```

```
struct karmasik {
float gercel;
float sanal;      };
struct karmasik toplam;
```

Temel Veri Yapıları

- Tüm programlama dillerinin, genel olarak, karakter, tamsayı, kesirli sayı ve sözcük (karakter katarı) saklanması için temel veri yapıları vardır. Veri yapısı, aslında, ham olarak 1 ve 0'lardan oluşan verinin yorumlanmasını belirleyen biçimleme (formating) düzenidir. Örneğin, 62 sayısının ikili tabandaki karşılığı, 111110 olarak bellekte saklanır.
- Temel veri yapıları aşağıdaki gibi sınıflanabilir:

- **Karakter** (Character) - A, B,C, @, ?, >,∞…
- **Tamsayı** (Integer) - 1923, 19, 29, 23, 5, - 153
- **Gerçel Sayı** (Real Number) - 3.14, 2.71, 1.53×10^8…
- **Katar** (String) - "Üniversite", "Oğuzhan", "Can", "A"
- **Dizi/Matris** (Array/Matrix) - [23 4 1923], $\begin{bmatrix} 19 & 23 \\ 29 & 5 \end{bmatrix}$

Tanımlamalı Veri Yapıları

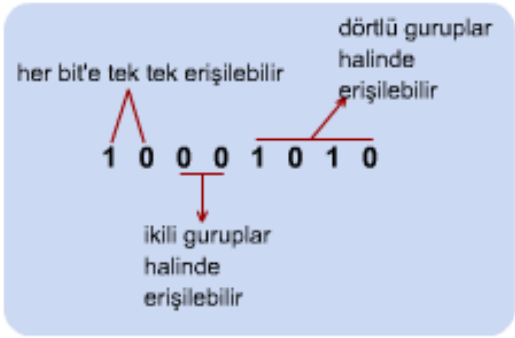
- Tanımlamalı veri yapısı, temel veya daha önceden tanımlanmış veri yapılarının kullanılıp yeni veri yapıları oluşturulmasıdır. Üç değişik şekilde yapılabilir:
  - **Topluluk (Struct) Oluşturma:** Birden çok veri yapısının bir araya getirilip yeni bir veri yapısı ortaya çıkarmaktır. (Java dilinde sınıflar)
  - **Ortaklık (Union) Oluşturma:** Birden çok değişkenin aynı bellek alanını kullanmasını sağlayan veri yapısı tanımlamasıdır. Ortaklıkta en fazla yer işgal eden veri yapısı hangisi ise, ortaklık içerisindeki tüm değişkenler orayı paylaşır.
  - **Bit Düzeyinde Erişim:** Verinin her bir bit'i üzerinde diğerlerinden bağımsız olarak işlem yapılması olanağı sunar.

- Her birinin kullanım amacı farklı farklı olup uygulamaya göre bir tanesi veya hepsi bir arada kullanılabilir. Genel olarak, en çok kullanılanı topluluk oluşturmadır; böylece birden fazla veri yapısı bir araya getirilip/paketlenip yeni bir veri yapısı/türü ortaya çıkarılır.

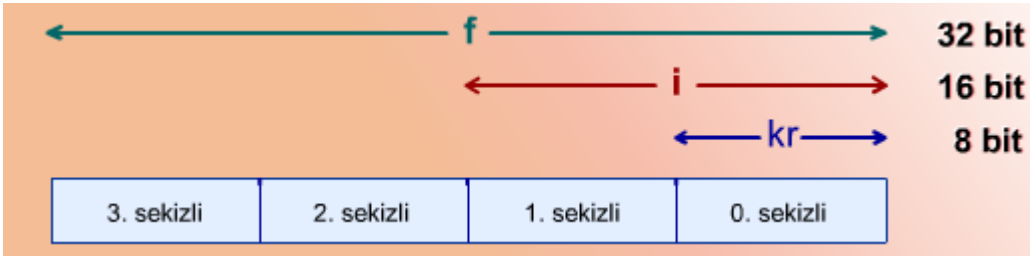
Tanımlamalı Veri Yapıları

- C dilinde tanımlamalı veri yapılarına örnek aşağıda verilmiştir.

```
struct kimlik {
   char ad[15];
   char soyad[20];
   int yas;
   char adres[50];
  };
```

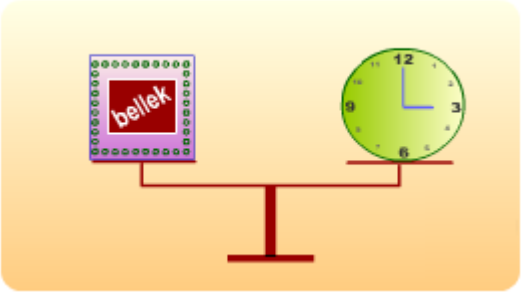



```
union paylas {
   int i;
   flat f;
   char kr;
  };_
```



Veri Modelleri

- Veri modelleri, tasarımı yapılacak programın en uygun ve etkin şekilde olmasını sağlar ve daha baştan programın çalışma hızı ve bellek gereksinimi hakkında bilgi verir. Çoğu zaman, programın çalışma hızıyla bellek gereksinimi miktarı doğru orantılıdır denilebilir.
- Veri modeller, genel olarak, aşağıdaki gibi verilebilir:
  - **Bağlantılı Liste (Link List)**
  - **Ağaç (Tree)**
  - **Graf (Graph)**
  - **Durum Makinası (State Machine)**
  - **Veritabanı-İlişkisel (Database Relational)**
  - **Ağ Bağlantı (Network Connection)**



Hız ile Bellek Miktarı arasında denge kurulması

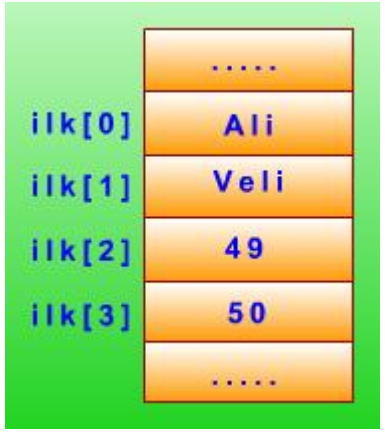
Liste ve Bağlantılı Liste Veri Modeli

- Liste veri modeli, aynı kümeye ait olan verilerin bellekte art arda tutulması ilkesine dayanır. Veriler belirli bir düzen içerisinde (sıralı vs.) olabilir veya olmayabilir; önemli olan tüm verilerin art arda gelen sırada tutulmasıdır.

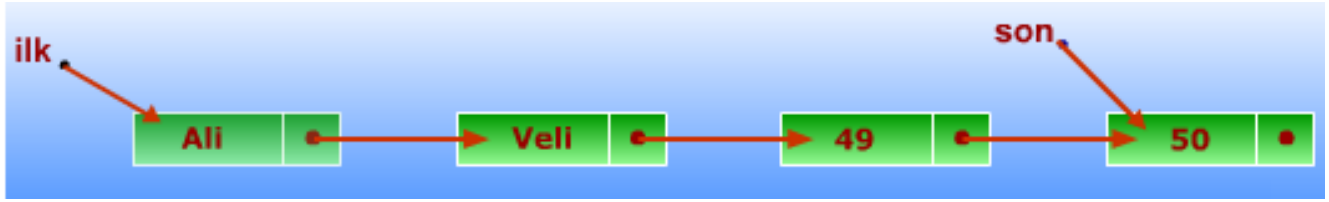


Liste ve Bağlantılı Liste Veri Modeli

- En yalın liste veri modeli bir boyutlu dizi üzerinde tutulanıdır. Böylesi bir listeye eleman ekleme işlemi oldukça kolaydır; genel olarak, yeni gelen elemanlar listenin sonuna eklenir. Yalın listede bir sonraki eleman hemen o elemanın işgal ettiği bellek alanından sonradır.



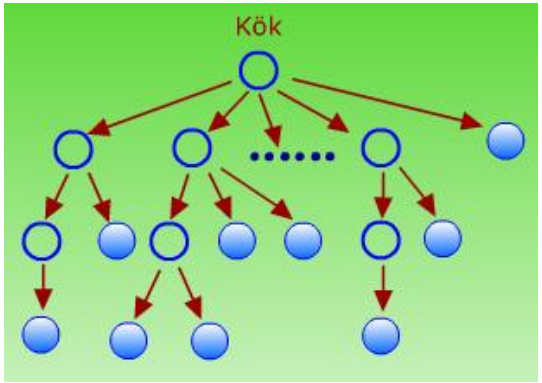
- Bağlantılı liste (link list) ise, elemanların kendi değerlerine ek olarak bir de bağlantı bilgisinin kullanılmasıyla sağlanır; bağlantı bilgisi bir sonraki elemanın adresi niteliğindedir.



Ağaç Veri Modeli

- Ağaç veri modeli, düğümlerden ve dallardan oluşur; düğümlerde verilerin kendileri veya bir kısmı tutulurken, dallar diğer düğümlere olan bağlantı ilişkilerini gösterir. Ağaç veri modeli, özellikle kümenin büyük olduğu ve arama işleminin çok kullanıldığı uygulamalarda etkin bir çözüm sunar.
- En üstteki düğüm kök (root), kendisine alttan hiçbir bağlantının olmadığı düğüm yaprak (leaf), diğerleri de ara düğüm (internal node) olarak adlandırılır. Bir düğüme alttan bağlı düğümlere çocuk (child), üsten bağlı düğüme de o düğümün ailesi (parent) denilir.

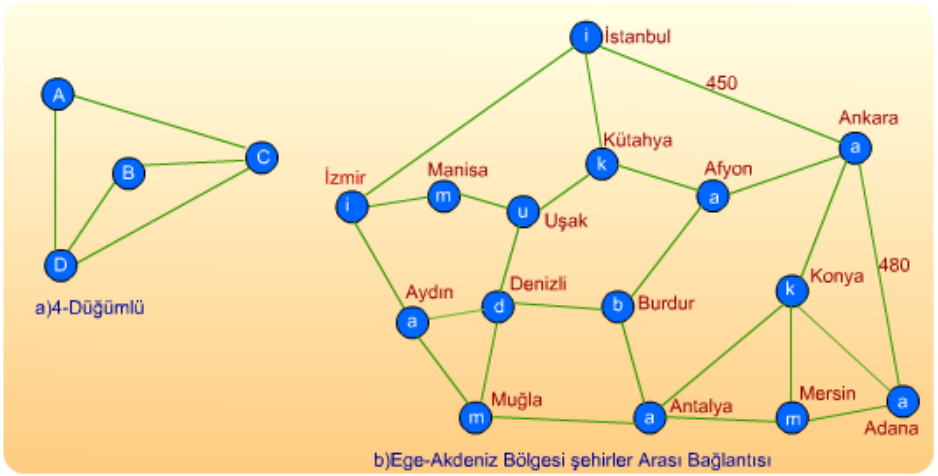
- Hızlı Erişim
- Esnek Bellek Kullanımı
- Arama-Ekleme Maliyetleri
- Tasarım Esnekliği
- Çok Farklı problemlere Model



Graf Veri Modeli

- Graf veri modeli, aynı kümeye ait olan verilerin şekilde görüldüğü gibi düğümler, ayrıtlar (kenarlar) ve bunların birleştirilmesinden oluşur. Düğümler birleşme noktasını ayrıtlar da düğümlerin bağlantı ilişkisini gösterir. Verilerin kendileri veya bir kısmı hem düğümlerde hem de ayrıtların bilgi kısmında tutulabilir.
  - Graflar, yazılım dünyasından önemli bir yere sahiptir. Örneğin, bir şehrin trafik altyapısından en yüksek akışın sağlanması, taşıma şirketinin en verimli taşıma şekli veya network bağlantılarında yüksek başarım elde edilmesi gibi problemler.

- Modelleme Esnekliği
- Üzerine Tasarlanmış Algoritma Çokluğu
- En kısa Yol (Shortest Path)
- Yol Ağacı (Spanning Tree)
- Greddy Yöntemi
- DFS ve BFS Yaklaşımları
- Kuruksal Algoritması



Durum Makinası Veri Modeli

- Durum makinası veri modeli, bir sistemin davranışını tanımlamak ve ortaya çıkarmak için kullanılan bir yaklaşım şeklidir; işletim sistemlerinde, derleyici ve yorumlayıcılarda, kontrol amaçlı yazılımlarda sistemin davranışını durumlara indirger ve durumlar arası geçiş koşullarıyla sistemi ortaya koyar.
  - Durum makinası, yazılım uygulamasında birçok alanda kullanılabilir. Örneğin bir robot kolunun hareketi, şifre çözme, gerçek zamanlı işletim sistemlerinde proses kontrolü ve genel olarak kontrol alt sistemlerinin yazılımla uygulamayı başarılı bir şekilde sonuçlandırma durumlarında çözüm olur.

Durum Makinası Veri Modeli

- Durum makinası veri modeli şeklen yönlü graflara benzer, ancak, birleşme noktaları graflarda olduğu gibi düğüm olarak değil de **durum**, ayrıtlar da **geçiş eğrileri** olarak adlandırılır. Durumlar arasındaki geçişler, sistemin o ana kadar ki durumlarına ve giriş parametrelerine bağlıdır.

- Davranış Modelleme
- Ardaşıl Yaklaşım
- Desen Uyuşması
- Katar Arama
- Gramer Çözümü

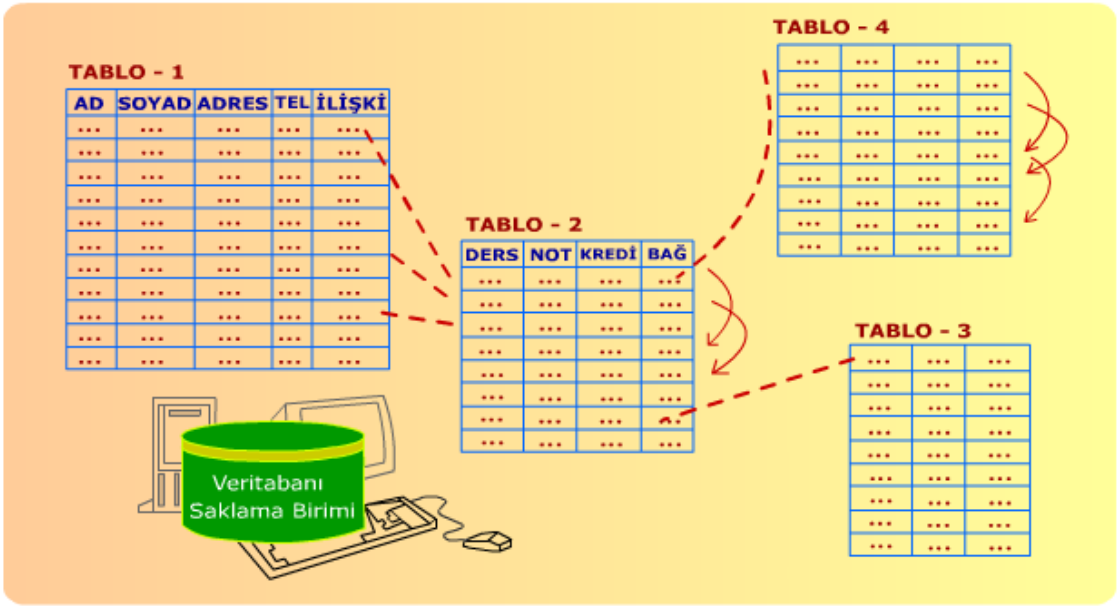


Aşk Sözcüğü İçin Durum Makinası

Veritabanında İlişkisel Veri Modeli

- Veritabanı ilişkisel veri modeli veritabanı uygulamalarında var olan dört beş sınıftan birisidir; veriler şekilde gösterildiği gibi tablolar üzerinden kurulan ilişkilere dayanmaktadır.

- Bilgilerin Düzenli Saklanması
- Hızlı Arama/Sorulama
- Veriler Arasında İlişki Oluşturulması
- İnternet Tarayıcısı Ortamında Bilgi Sorgulama
- Koruma
- Verilerin Arşivlenmesi



Yukarıda tipik olarak veritabanı ilişkisel veri modelindeki tablolar ve alanların durumu görülmektedir.

Veritabanında İlişkisel Veri Modeli

- SQL (Structured Query Language), sorgulama dili kullanılarak veritabanı üzerinde sorgulama yapılabilir. Access, Microsoft SQL, ORACLE, SYBASE, Ingres gibi birçok veritabanı yönetim sistemleri ilişkisel veri modelini desteklemektedir.

- Veritabanı yönetim sistemleri, veritabanı oluşturma, tablo yaratma, alanları tanımlama gibi işlerin başarılı bir şekilde sonuçlandırmasını ve genel olarak veritabanı yönetimini sağlarlar.

Ağ Veri Modeli

- Ağ veri modeli, katmalı ağ mimarilerinde, bilgisayarlar arasında eş katmanlar (peer layers) düzeyinde veri alış-verişini sağlayan dilim (segment), paket (packet) ve çerçeve yapılarını ortaya koyar ve iletişim için gerekli davranışı tanımlar. Veri haberleşmesinde hemen hemen tüm mimariler katmanlı yapıdadır. Tüm mimariler örnek temsil eden OSI 1, başvuru modeli 7, TCP/IP (**T**ransmission **C**ontrol **P**rotocol / **I**nternet **P**rotocol) protokol kümesi 4 katmanlıdır.



Veri Modelleri

- Bu derste aşağıdaki veri modelleri detaylı ele alınacaktır;
- **Liste**
  - Sonlu sayıda elemandan oluşan ve elemanları doğrusal sırada yerleştirilmiş veri modeli. Herhangi bir elemanına erişimde sınırlama yoktur.
- **Yığıt veya Yığın**
  - Elemanlarına erişim sınırlaması olan, liste uyarlı veri modeli (Last In First Out-LIFO listesi).
- **Kuyruk**
  - Elemanlarına erişim sınırlaması olan, liste uyarlı veri modeli. (First In First Out-FIFO listesi).
- **Ağaç**
  - Doğrusal olmayan belirli niteliklere sahip veri modeli
- **Çizge (Graph)**
  - Köşe adı verilen düğümleri ve kenar adı verilen köşeleri birbirine bağlayan bağlantılardan oluşan doğrusal olmayan veri modeli

Veri Yapısı	Artıları	Eksileri
Dizi	Hızlı ekleme ve çok hızlı erişim(indis biliniyorsa).	Yavaş arama, yavaş silme ve sabit boyut.
Sıralı Dizi	Sıralanmamış diziye göre daha hızlı arama.	Yavaş arama, yavaş silme ve sabit boyut.
Yığın	Son giren, ilk çıkar(last-in, first-out) erişimi sağlar.	Diğer öğelere yavaş erişim.
Kuyruk	İlk giren, ilk çıkar(first-in, first-out) erişimi sağlar.	Diğer öğelere yavaş erişim.
Bağlı Liste	Hızlı ekleme ve silme.	Yavaş arama.
Hash Tablosu	Hızlı ekleme ve anahtar bilindiğinde çok hızlı erişim.	Yavaş silme, anahtar bilinmediğinde yavaş erişim ve verimsiz bellek kullanımı.
Küme(Heap)	Hızlı ekleme ve silme.	Diğer öğelere yavaş erişim. Başta en büyük öğeye erişim.
İkili Ağaç	Hızlı arama, ekleme ve silme(ağaç dengeli kalmışsa).	Silme algoritması karmaşık.
Graf	Gerçek-dünya problemlerini modelleyebilmesi.	Bazı algoritmaları yavaş çalışmakta ve karmaşıklığı yüksek.

Veri Yapıları- Bölüm Özeti

- Veri modelleri ve onlara ait veri yapıları yazılım geliştirmenin temel noktalarıdır; problemlerin en etkin şekilde çözülebilmesi için ona algoritmik ifadenin doğasına yakın bulunmasıdır. Kısaca, veri yapıları, verinin saklanma kalıbını, veri modelleri de veriler arasındaki ilişkiyi gösterir.

- Bilinen ve çözümlerde sıkça başvurulan veri modelleri, genel olarak, bağlantılı liste (link list), ağaç (tree), graf (graph), durum makinası (state machine), ağ (network) ve veritabanı-ilişkisel (database-relation) şeklinde verilebilir.

- Her veri modelinin, altında duran veri yapısına bağlı olarak, işlem zaman maliyetleri ve bellek gereksinimleri farklıdır. Program geliştirilirken, zaman ve bellek alanı maliyetlerini dengeleyecek çözüm üretilmeye çalışılır. Genel olarak, RAM türü ardışıl erişimlerin yapılabildiği bellek üzerinde, maliyeti ile bellek gereksinim ters orantılı olduğu söylenebilir.

Algoritmik Program Tasarımı ve Analizi

Bölüm 2

Algoritmik Program Tasarımı Nedir?

- Verilen bir problemin bilgisayar ortamında çözülecek biçimde adım adım ortaya koyulması ve herhangi bir programlama aracıyla kodlanması sürecidir.
- Çözüm için yapılması gereken işlemler hiçbir alternatif yoruma izin vermeksizin sözel olarak ifade edilir.
- Verilerin, bilgisayara hangi çevre biriminden girileceğinin, problemin nasıl çözüleceğinin, hangi basamaklardan geçirilerek sonuç alınacağının, sonucun nasıl ve nereye yazılacağının sözel olarak ifade edilmesi biçiminde de tanımlanabilir.

Algoritmanın Önemli Özellikleri

- Algoritma hazırlanırken, çözüm için yapılması gerekli işlemler, öncelik sıraları göz önünde bulundurularak ayrıntılı bir biçimde tanımlanmalıdırlar.
- Yazılan komutun tek bir anlama gelmesi ve herkes tarafından anlaşılır olması gereklidir.
- Yazılan komutların uygulanabilir olması gereklidir.
- Her algoritmanın sonlanması, çalıştırılan komut sayısının sonsuz olmaması gereklidir.

Algoritma Süreci

- Tasarım (design)
- Doğruluğunu ispat etme (validation)
- Analiz (analysis)
- Uygulama (implementation)
- Test

Kaba-Kod (Pseudo Code)

- Kaba-kod, bir algoritmanın yarı programlama kuralı, yarı konuşma diline dönük olarak ortaya koyulması, tanımlanması, ifade edilmesidir.

- Kaba-kod, çoğunlukla, bir veri yapısına dayandırılmadan algoritmayı genel olarak tasarlamaya yardımcı olur.

Gerçek Kod

- Algoritmanın herhangi bir programlama diliyle, belirli bir veri yapısı üzerinde gerçekleştirilmiş halidir.
- Bir algoritmanın gerçek kodu, yalnızca, tasarlandığı veri yapısı üzerinde çalışır.
- Bir algoritma kaba-kod ile verilirse gerçek kod verilmesin den daha kolay anlaşılır.

Kaba-kod: temel gösterim

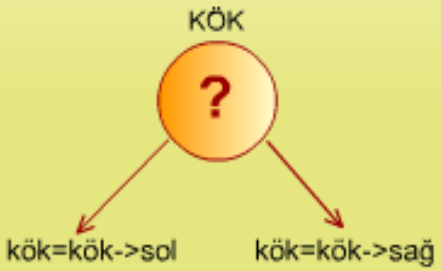
- 1. Bir değer atamak için genellikle **:=** kullanılır. = işareti ise eşitlik kontrolü için kullanılır.
- 2. Metot, fonksiyon, yordam isimleri:
  Algoritma Adı ({parametre listesi})
- 3. Program yapısı şu şekilde tanımlanır::
  - Karar yapıları: **if ... then ... else ...**
  - while döngüleri: **while ... do {döngü gövdesi}**
  - Tekrar döngüleri: **repeat {döngü gövdesi} until ...**
  - for döngüleri: **for ... do {döngü gövdesi}**
  - Dizi indeksleri: **A[i]**
- 4. Metotların çağrılması: **Metot adı ({değişken listesi})**
- 5. Metotlardan geri dönüş: **return değer**

Algoritmaların kaba-kod olarak ifade edilmesi

- Örnek: Bir dizideki elemanların toplam ve çarpımını hesaplayan algoritmayı kaba-kod kullanarak tanımlayınız.
  - **Toplam Ve Çarpım Hesapla** (dizi, toplam, çarpım)
  - **Girdi:** n sayıdan oluşan dizi.
  - **Çıktı:** dizi elemanlarının toplam ve çarpım sonucu
  - **for** i:= 1 **to** n **do**
  - toplam:= toplam + dizi[i] toplam: toplam + dizi[i]
  - çarpım:= çarpım* dizi[i]
  - **endfor**

Kaba-kod ve Gerçek Kod

```
/* İkili arama ağacına düğüm ekleme algoritmasının kaba-kodu */
if (kök boş ağacı gösteriyorsa)
   Düğümü köke ekle;
else {
   if (eklenen kökten küçükse)
    sol altağaca dallan; (kök=kökün sol çocuğu adresi )
    başa dön;
   else
    sağ altağaca dallan; (kök=kökün sağ çocuğu adresi )
    başa dön;
   }
```




```
/* Ikili arama agacina dügüm ekleme fonksiyonu */
void ekle (agacKok, eklenen )
            AGAC2 *agacKok, *eklenen;
{
  if (agacKok==NULL )
     kok=eklenen;
     else {
        if (eklenen->bilgi <= agacKok->bilgi)
          {
           if (agacKok->sol==NULL)
              agacKok->sol=eklenen;
           else ekle (agacKok->sol, eklenen );
          }
        else
          {
             if (agacKok->sag==NULL )
                agacKok->sag=eklenen;
             else ekle (agacKok->sag, eklenen );
          }
      }
}
```

Algoritma Analizi

- Algoritma analizi, tasarlanan program veya fonksiyonun belirli bir işleme göre matematiksel ifadesini bulmaya dayanır.
- Burada temel hesap birimi seçilir ve programın görevini yerine getirebilmesi için bu işlemden kaç adet yapılması gerektiğini bulmaya yarayan bir bağıntı hesaplanır.
- Eğer bu bağıntı zamanla ilgiliyle çalışma hızını, bellek gereksinimiyle ilgiliyse bellek gereksinimi ortaya koyar.

Algoritma Analizi

- Neden algoritmayı analiz ederiz?
  - Algoritmanın performansını ölçmek için
  - Farklı algoritmalarla karşılaştırmak için
  - Daha iyisi mümkün mü? Olabileceklerin en iyisi mi?
- Algoritmayı nasıl analiz ederiz?
  - Yürütme zamanı(Running Time)-T(n)
  - Karmaşıklık (Complexity) -Notasyonlar

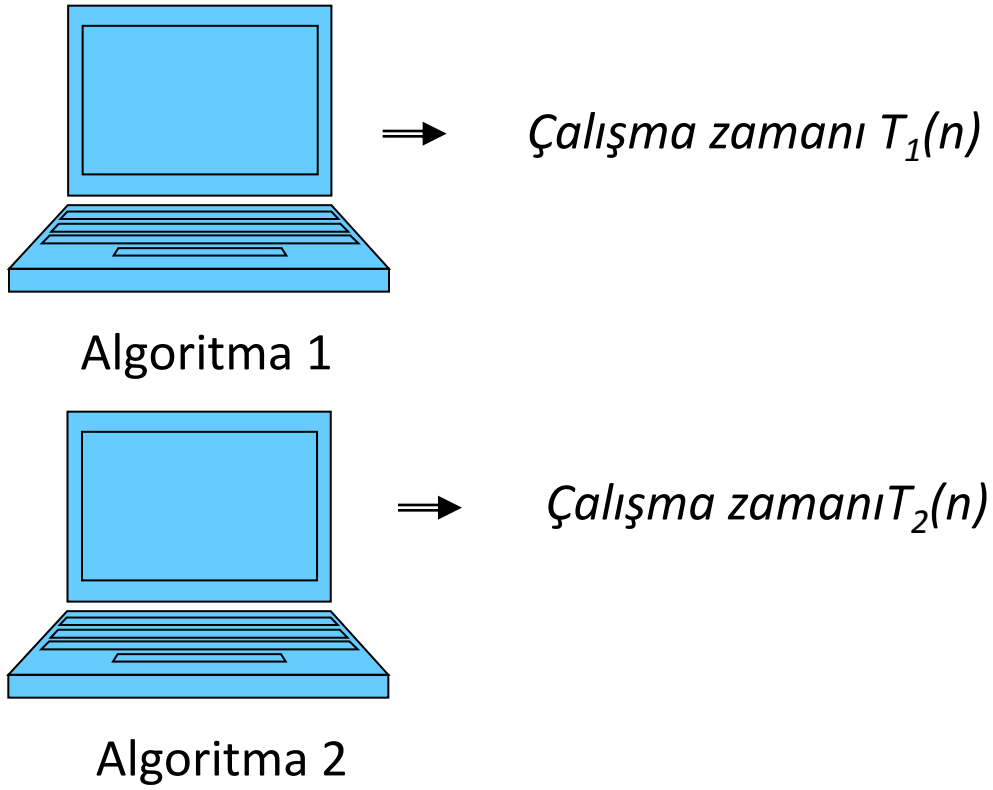
Yürütme Zamanı Analizi (Running Time)

- **Yürütme Zamanı**; Bir programın veya algoritmanın işlevini yerine getirebilmesi için, temel kabul edilen işlevlerden kaç adet yürütülmesini veren bir bağıntıdır ve T(n) ile gösterilir.

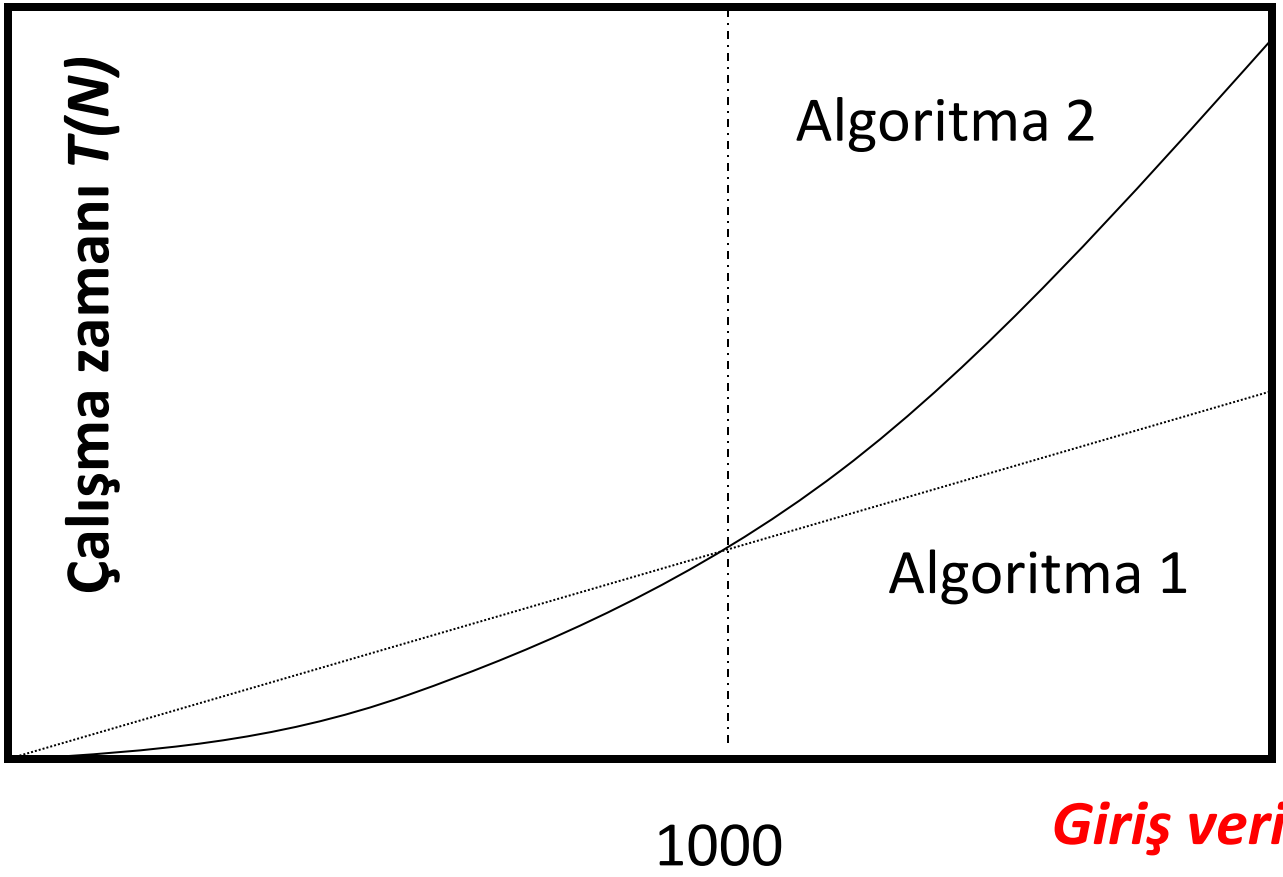
- Temel hesap birimi olarak, programlama dilindeki deyimler seçilebildiği gibi döngü sayısı, toplama işlemi sayısı, atama sayısı, dosyaya erişme sayısı gibi işler de temel hesap birimi olarak seçilebilir.

Yürütme Zamanı Analizi

- Algoritma 1, T1(N)=1000N
- Algoritma 2, T2(N)=N^2



Yürütme (Çalışma) Zamanı Analizi



Çalışma Zamanlarının Özeti

N	T1	T2
10	10^{-2} sec	10^{-4} sec
100	10^{-1} sec	10^{-2} sec
1000	1 sec	1 sec
10000	10 sec	100 sec
100000	100 sec	10000 sec

N değerinin 1000'den küçük olduğu durumlarda iki algoritma arasındaki çalışma zamanı ihmal edilebilir büyüklüktedir.

Analiz

- Çalışma zamanının kesin olarak belirlenmesi zordur
  - **Giriş verilerine bağlı olan en iyi durum** (best case)
  - **Ortalama durum** (average case); hesaplanması zordur
  - **Diğerlerine göre en kötü durum** (worst case); hesaplanması kolaydır
- Bunun için çeşitli notasyonlardan faydalanılır.

Aritmetik Ortalama için *T(n)* Hesabı

- **Örnek 1:** Aşağıda bir dizinin aritmetik ortalamasını bulan ve sonucu çağırana gönderen **bulOrta()** fonksiyonun kodu verilmiştir. Bu fonksiyonun yürütme zamanını gösteren **T(n)** bağıntısını ayrık C dili deyimlerine göre belirleyiniz.

```
        float bulOrta(float A[], int n) {
        {
        float ortalama, toplam=0;
        int k ;
1-      for(k=0;k<n;k++)
2-              toplam+=A[k];
3-      ortalama=toplam/n
4-      return ortalama;
        }
```

Aritmetik Ortalama için *T(n)* Hesabı

- **Çözüm 1:**

Temel Hesap Birimi	Birim Zaman (Unit Time)	Frekans(Tekrar) (Frequency)	Toplam (Total)
float bulOrta(float A[], int n)	-	-	-
{	-	-	-
float ortalama, toplam=0;	-	-	-
int k ;	-	-	-
1- for(k=0;k<n;k++)	1,1,1	1, (n+1), n	2n+2
2- toplam+=A[k];	1	n	n
3- ortalama=toplam/n	1	1	1
4- return ortalama;	1	1	1
}	-	-	-
			T(n)=3n+4

En Küçük Eleman Bulma için *T(n)* Hesabı

- **Örnek 2:** Aşağıda bir dizi içerisindeki en küçük elamanı bulan **bulEnKucuk()** adlı bir fonksiyonun kodu görülmektedir. Bu fonksiyonun yürütme zamanını gösteren *T(n)* bağıntısı ayrık C dili deyimlerine göre belirleyiniz.

```
        float bulEnKucuk(float  A[])
        {
        float enkucuk;
        int k ;
1-      enkucuk=A[0];
2-      for(k=1;k<n;k++)
3-              if (A[k]<enkucuk)
4-                      enkucuk=A[k];
5-      return enkucuk;
        }
```

En Küçük Eleman Bulma için *T(n)* Hesabı

- **Çözüm 2:**

Temel Hesap Birimi	Birim Zaman (Unit Time)	Frekans(Tekrar) (Frequency)	Toplam (Total)
float bulEnKucuk(float A[]) {	-	-	-
float enkucuk;	-	-	-
int k ;	-	-	-
1- enkucuk=A[0];	1	1	1
2- for(k=1;k<n;k++)	1,1,1	1, n, (n-1)	2n
3- if (A[k]<enkucuk)	1	n-1	n-1
4- enkucuk=A[k];	1	n-1	n-1
5- return enkucuk;	1	1	1
}	-	-	-
			T(n)=4n

Matris Toplama için *T(n)* Hesabı

Temel Hesap Birimi	Birim Zaman (Unit Time)	Frekans(Tekrar) (Frequency)	Toplam (Total)
void toplaMatris (A,B,C) {	-	-	-
int A[n][m], B[n][m], C[n][m];	-	-	-
int i,j ;	-	-	-
1- for(i=0;i<n;i++)	1,1,1	1,(n+1),n	2n+2
2- for(j=0;j<m;j++)	1,1,1	n(1,(m+1),m)	n(2m+2)
3- C[i][j]=A[i][j]+B[i][j];	1	nm	nm
}	-	-	-
			T(n,m)=3nm+4n+2
n=m ise			$T(n)=3n^2+4n+2$

$$T(N)=\sum_{i=1}^{N}\sum_{j=1}^{N}1=\sum_{i=1}^{N}N=N*N=N^2$$

Karmaşıklık (Copmlexity)

- Karmaşıklık; bir algoritmanın çok sayıda parametre karşısındaki değişimini gösteren ifadelerdir. Çalışma (Yürütme) zamanını daha doğru bir şekilde bulmak için kullanılırlar.

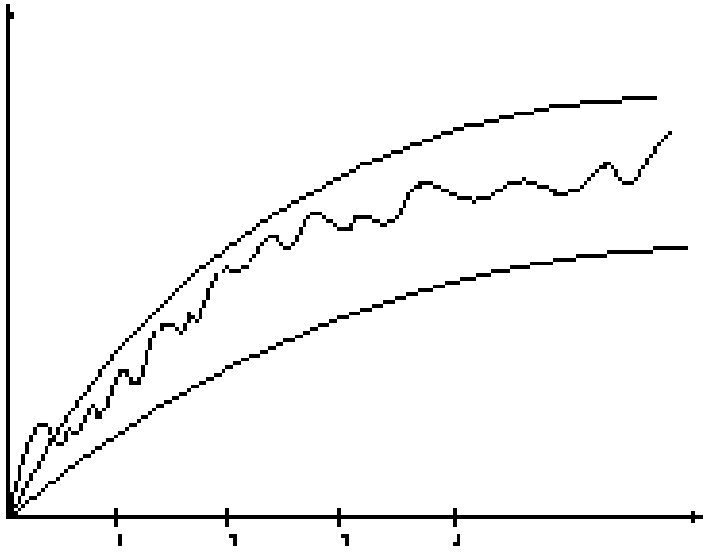
- Genel olarak, az sayıda parametreler için karmaşıklıkla ilgilenilmez; eleman sayısı ***n***'nin sonsuza gitmesi durumunda algoritmanın maliyet hesabının davranışını görmek veya diğer benzer işleri yapan algoritmalarla karşılaştırmak için kullanılır.

- Karmaşıklığı ifade edebilmek için asimtotik ifadeler kullanılmaktadır.

- Bu amaçla O(n) (O notasyonu), Ω(n) (Omega notasyonu), θ(n) (Teta notasyonu) gibi tanımlamalara baş vurulur.

Karmaşıklık (Copmlexity)

- Strateji: Alt ve üst limitlerin bulunması



Üst limit – O(n)

Algoritmanın gerçek fonksiyonu

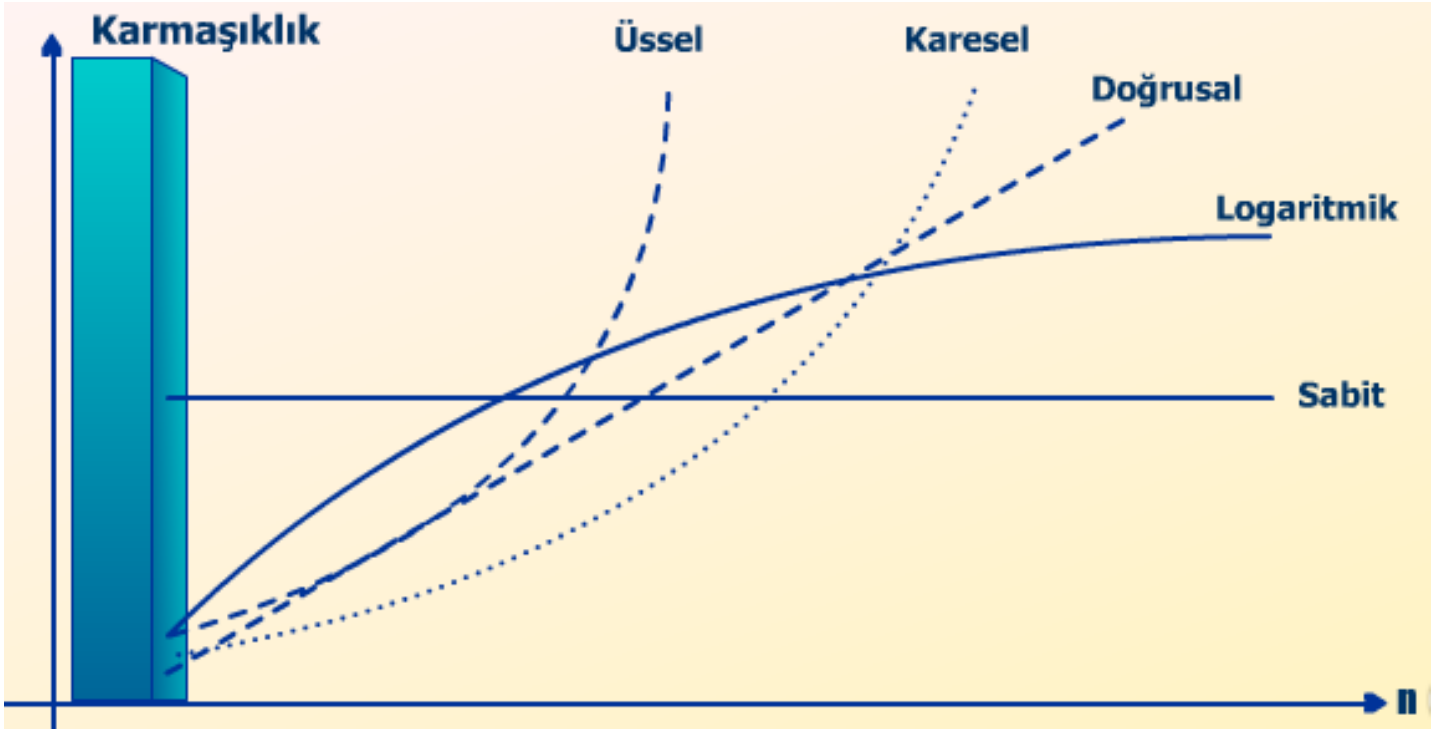
Alt limit- Ω (n)

Karşılaşılan Genel Fonksiyonlar

Maliyet artar

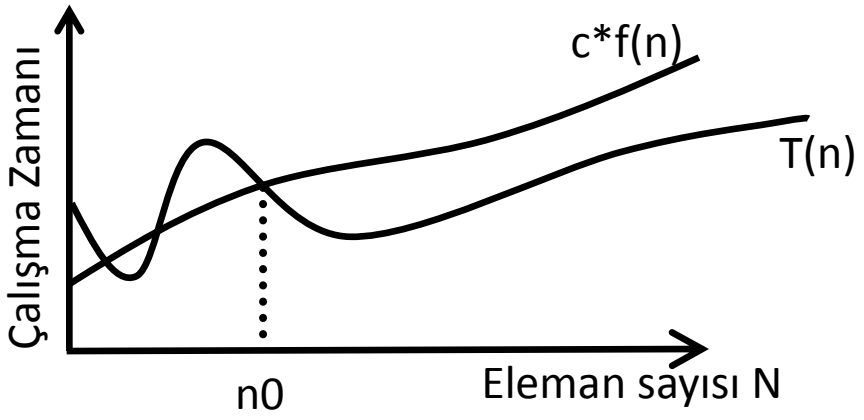
Büyük O	Değişim Şekli
$O(1)$	Sabit, komut bir veya birkaç kez çalıştırılır. Yenilmez!
$O(\log_n)$	Logaritmik, Büyük bir problem, her bir adımda sabit kesirler tarafından orijinal problemin daha küçük parçalara ayrılması ile çözülür. İyi hazırlanmış arama algoritmalarının tipik zamanı
$O(n)$	Doğrusal, Küçük problemlerde her bir eleman için yapılır. Hızlı bir algoritmadır. N tane veriyi girmek için gereken zaman.
$O(n\log_n)$	Doğrusal çarpanlı logaritmik. Çoğu sıralama algoritması
$O(n^2)$	Karasel. Veri miktarı az olduğu zamanlarda uygun (N<1000)
$O(n^3)$	Kübik. Veri miktarı az olduğu zamanlarda uygun (N<1000)
$O(2^n)$	İki tabanında üssel.Veri miktarı çok az olduğunda uygun (N<=20)
$O(10^n)$	On tabanında üssel
$O(n!)$	Faktöriyel

Karmaşıklık (Copmlexity)



Büyük-Oh(Big-Oh) Notasyonu: Asimptotik Üst Sınır (En kötü durum analizi)

- Bir algoritmanın çalışma süresi,
- T(N)=O(f(n)) **O**, bir fonksiyon değil, sadece gösterimdir.
- ***T(N)* ≤ *c f(n)*** *ve* ***N* ≥ *n*$_0$** *koşullarını sağlayan* ***c*** *ve* ***n*$_0$** *değerleri varsa* ***T(N)* ≤ *c f(n)*** *ifadesi doğrudur.*
- *f(n), T(N)'in asimtotik üst limiti olarak adlandırılır.*
- *T(N)=O(f(n))*



O Notasyonu- Asimtotik Üst Limit

A algoritması için $T_A(N) = 1000N$ olarak verilmiştir. O notasyonu cinsinden bu çalışma süresi $T_A(N) = 1000N = O(N)$ dir.

Yukarıdaki ifadenin doğruluğunu ispat etmek için O notasyonunun tanımını kullanacağız. Bu tanıma göre $1000N = O(N)$ ifadesini ispatlamak için,

$1000N \le cN \qquad \forall N \ge n_0$ eşitsizliğini belirli bir c ve n_0 sabitleri için ispatlamak gerekir.

$c = 2000$ ve $n_0 = 1$ seçildiği zaman eşitsizliğin n_0 dan büyük her N değeri için sağlandığı aşikardır.

O Notasyonu- Asimtotik Üst Limit

B algoritması için $T_B(N) = N^2$ olarak verilmiştir. O notasyonu cinsinden bu çalışma süresi $T_B(N) = N^2 = O(N^2)$ dir.

Yukarıdaki ifadenin doğruluğunu ispat etmek için O notasyonunun tanımını kullanacağız. Bu tanıma göre $N^2 = O(N^2)$ ifadesini ispatlamak için,

$N^2 \leq cN^2 \qquad \forall N \geq n_0$ eşitsizliğini belirli bir c ve n_0 sabitleri için ispatlamak gerekir.

$c = 1$ ve $n_0 = 1$ seçildiği zaman eşitsizlik n_0 dan büyük her N değeri için sağlanmaktadır.

O Notasyonu- Asimtotik Üst Limit

$7n^2 + 5 = O(n)$ ifadesinin doğru olup olmadığını ispatlayın.

O notasyonuna göre

$7n^2 + 5 \le cn \quad \forall n \ge n_0$ olması gereklidir. Bu şartı sağlayacak c ve n_0 değerleri bulabilir miyiz?

Her iki tarafı n sayısına bölersek;

$7n + \frac{5}{n} \le c$ elde edilecektir. Buna göre eşitliğin sağlanabilmesi için n sayısı değiştikçe c de değişmelidir. Sabit bir c ve n_0 çifti yoktur; dolayısıyla eşitsizlik doğru değildir.

O Notasyonu

$7n^2 + 5 = O(n^2)$ ifadesinin doğruluğunu ispatlayın.

O notasyonu tanımına göre,

$7n^2 + 5 \leq cn^2$ $\forall n \geq n_0$ ifadesini sağlayan bir c ve n_0 değeri var mı?

$c = 12$ $n_0 = 1$ veya

$c = 8$ $n_0 = 5$ değerleri eşitsizliği n_0 dan büyük her n değeri için sağlamaktadır.

Not: Çözüm kümesini sağlayan kaç tane c ve n çifti olduğu önemli değildir. Tek bir çift olması notasyonun doğruluğunu ispatlamak için yeterlidir.

Büyük-Oh(Big-Oh) Notasyonu: Asimptotik Üst Sınır

- Örnek: T(n) = 2n+5 is $O(n^2)$ Neden?
  - $n>=n_0$ şartını sağlayan tüm sayılar için $T(n) = 2n+5 <= c*n^2$ şartını sağlayan c ve n_0 değerlerini arıyoruz.
  - $n>=4$ için $2n+5 <= 1*n^2$
    - c = 1, no = 4
  - $n>=3$ için $2n+5 <= 2*n^2$
    - c = 2, no = 3
  - Diğer c ve n_0 değerleri de bulunabilir.

Büyük-Oh(Big-Oh) Notasyonu: Asimptotik Üst Sınır

- Örnek: T(n) = n(n+1)/2 → O(?)
  - T(n) = $n^2/2 + n/2$ → $O(N^2)$. Neden?
  - n >= 1 iken $n^2/2 + n/2 <= n^2/2 + n^2/2 <= n^2$
  - Böylece, T(n)=n*(n+1)/2 <= 1* n^2 for all n >=1
    - c=1, no=1
  - Not: T(n) ayrıca $O(n^3)$ tür.

O Notasyonunun Önemi

- İki algoritma karşılaştırılırken zaman mertebesinden konuşulur. Mertebesi büyük olanın daha yavaş olduğu kolaylıkla anlaşılır.

- Makineler arasındaki fark, katsayılar olarak düşünüldüğünde $O(7n^2)$ yerine $O(n^2)$ ifadesi geçerli olur. Dolayısıyla sabitler ihmal edilebilir ve birimlerden kurtulur

O Notasyonu

- O notasyonunda yazarken en basit şekilde yazarız.
  - Örneğin
    - $3n^2+2n+5 = O(n^2)$
  - Aşağıdaki gösterimlerde doğrudur fakat kullanılmaz.
    - $3n^2+2n+5 = O(3n^2+2n+5)$
    - $3n^2+2n+5 = O(n^2+n)$
    - $3n^2+2n+5 = O(3n^2)$

O Notasyonu-Örnek 1

- $3n^2+2n+5 = O(n^2)$ ifadesinin doğru olup olmadığını ispatlayınız.

$10\ n^2 = 3n^2 + 2n^2 + 5n^2$

$\geq 3n^2 + 2n + 5$ için $n \geq 1$

$c = 10,\ n_0 = 1$

Çözüm kümesini sağlayan kaç tane n_0 ve c cifti olduğu önemli değildir. Tek bir çift olması notasyonun doğruluğu için yeterlidir.

O Notasyonu-Örnek 2

- T(N)=O($7n^2+5n+4$) olarak ifade edilebiliyorsa, T(N) fonksiyonu aşağıdakilerden herhangi biri olabilir.
- T(N)=n^2
- T(N)=4n+7
- T(N)=$1000n^2+2n+300$

- T(N)= O($7n^2+5n+4$) =O(n^2)

O notasyonu- Örnek 3

- Fonksiyonların harcadıkları zamanları O notasyonuna göre yazınız.

- $f1(n) = 10\ n + 25\ n^2$
- $f2(n) = 20\ n \log n + 5\ n$
- $f3(n) = 12\ n \log n + 0.05\ n^2$
- $f4(n) = n^{1/2} + 3\ n \log n$

- $O(n^2)$
- $O(n \log n)$
- $O(n^2)$
- $O(n \log n)$

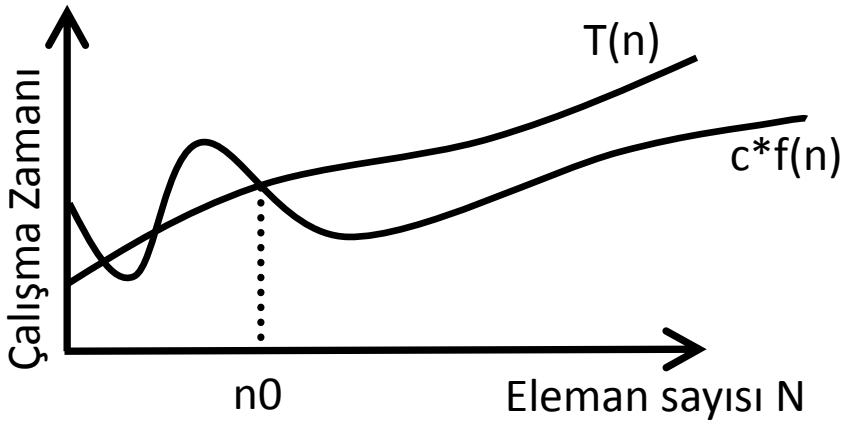
- $2n^2 = O(n^3)$: $2n^2 \leq cn^3 \Rightarrow 2 \leq cn \Rightarrow c = 1$ and $n_0 = 2$
- $n^2 = O(n^2)$: $n^2 \leq cn^2 \Rightarrow c \geq 1 \Rightarrow c = 1$ and $n_0 = 1$
- $1000n^2+1000n \neq O(n^2)$: Şart doğrumu

- $n = O(n^2)$:

$n \leq cn^2 \Rightarrow cn \geq 1 \Rightarrow c = 1$ and $n_0 = 1$

Ω Notasyonu- Asimtotik Alt Limit (En iyi durum analizi)

- O notasyonun tam tersidir.
- Her durumda **T(N) ≥ c f(n)** ve **N ≥ n_0** koşullarını sağlayan pozitif, sabit **c** ve **n_0** değerleri bulunabiliyorsa **T(N)=Ω(f(n))** ifadesi doğrudur.
- f(n), T(N)'in asimtotik alt limiti olarak adlandırılır.



Ω notasyonu-Örnek

- T(n) = 2n + 5 → Ω(n). Neden?
  - 2n+5 >= 2n, tüm n >= 1 için
- T(n) = 5*n^2 - 3*n → Ω(n^2). Neden?
  - 5*n^2 - 3*n >= 4*n^2, tüm n >= 4 için

- $7n^2+3n+5 = O(n^4)$
- $7n^2+3n+5 = O(n^3)$
- $7n^2+3n+5 = O(n^2)$
- $7n^2+3n+5 = \Omega(n^2)$
- $7n^2+3n+5 = \Omega(n)$
- $7n^2+3n+5 = \Omega(1)$

Examples

- **$5n^2 = \Omega(n)$**

  $\exists\ c, n_0$ such that: $0 \le cn \le 5n^2 \Rightarrow$ $cn \le 5n^2 \Rightarrow c = 1$ and $n_0 = 1$

- **$100n + 5 \neq \Omega(n^2)$**

  $\exists\ c, n_0$ such that: $0 \le cn^2 \le 100n + 5$

  $100n + 5 \le 100n + 5n\ (\forall\ n \ge 1) = 105n$

  $cn^2 \le 105n \Rightarrow n(cn - 105) \le 0$

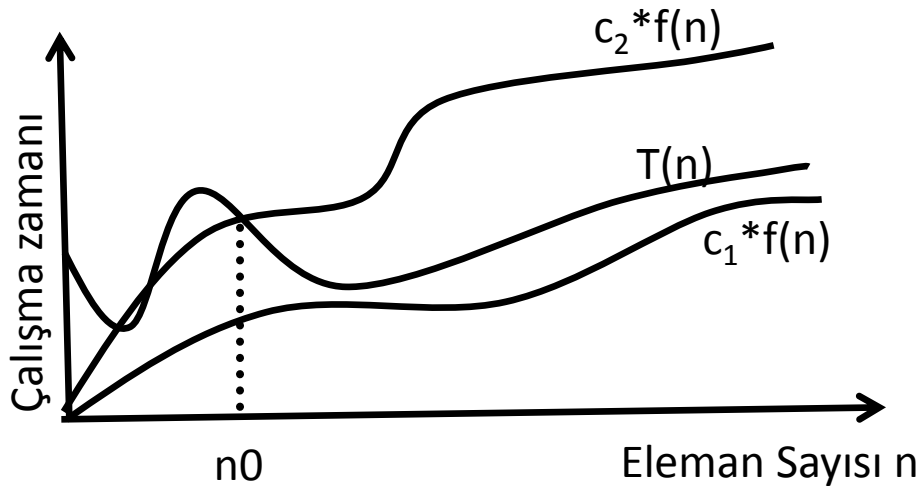
  Since n is positive $\Rightarrow cn - 105 \le 0 \Rightarrow n \le 105/c$

  $\Rightarrow$ contradiction: n cannot be smaller than a constant

- **$n = \Omega(2n)$, $n^3 = \Omega(n^2)$, $n = \Omega(\log n)$**

Θ Notasyonu (Ortalama durum analizi)

- Her durumda $c_1 f(n) \geq T(n) \geq c_2 f(n)$ ve $n \geq n_0$ koşullarını sağlayan pozitif, sabit c_1,c_2 ve n_0 değerleri bulunabiliyorsa $T(n)= \Theta(f(n))$ ifadesi doğrudur.



Θ notasyonu- Örnek

- $T(n) = 2n + 5 \rightarrow \Theta(n)$. Neden?

  $2n <= 2n+5 <= 3n$, tüm $n >= 5$ için

- $T(n) = 5*n^2 - 3*n \rightarrow \Theta(n^2)$. Neden?
  - $4*n^2 <= 5*n^2 - 3*n <= 5*n^2$, tüm $n >= 4$ için

Examples

- $n^2/2 - n/2 = \Theta(n^2)$
  - $\frac{1}{2}\, n^2 - \frac{1}{2}\, n \leq \frac{1}{2}\, n^2 \; \forall n \geq 0 \quad \Rightarrow \quad c_2 = \frac{1}{2}$
  - $\frac{1}{2}\, n^2 - \frac{1}{2}\, n \geq \frac{1}{2}\, n^2 - \frac{1}{2}\, n * \frac{1}{2}\, n \; (\forall n \geq 2) = \frac{1}{4}\, n^2 \Rightarrow \quad c_1 = \frac{1}{4}$

- $n \neq \Theta(n^2)$: $c_1\, n^2 \leq n \leq c_2\, n^2 \Rightarrow$ only holds for: $n \leq 1/c_1$
- $6n^3 \neq \Theta(n^2)$: $c_1\, n^2 \leq 6n^3 \leq c_2\, n^2 \Rightarrow$ only holds for:

  $n \leq c_2 / 6$

N değerini keyfi olarak belirlemek imkansızdır. Çünkü c_2 sabittir.

Another example

- Prove that $\frac{1}{2}n^2 - 3n = \Theta(n^2)$
  - **Determine c_1, c_2 and n_0 such that**

$$c_1 n^2 \le \frac{1}{2}n^2 - 3n \le c_2 n^2$$

$$c_1 \le \frac{1}{2} - \frac{3}{n} \le c_2$$

$$\frac{1}{2} - \frac{3}{n} \le c_2 \rightarrow n \ge 1, c_2 \ge \frac{1}{2}$$

$$c_1 \le \frac{1}{2} - \frac{3}{n} \rightarrow n \ge 7, c_1 \le \frac{1}{14}$$

For any polynomial $p(n) = \sum_{i=0}^{d} a_i n^i$
p(n) = $\Theta(n^d)$

  - c_1=1/14, c_2=1/2, n_0=7

Büyük-Oh, Theta, Omega

- İpucu:
- O(f(N)) düşünürsek f(N) ile “eşit veya küçük”
  - Üstten sınır: f(N) ile “yavaş veya aynı hızda büyür”
- Ω(f(N)) düşünürsek f(N) ile “eşit veya büyük”
  - Alttan sınır: f(N) ile “aynı hızda veya hızlı büyür”
- Θ(f(N)) düşünürsek f(N) ile “eşit”
  - Alttan ve Üsten sınır : büyüme oranları eşit
- (N’nin büyük olduğu ve sabitlerin elendiği durumlarda)

Sıkça Yapılan Hatalar

- Karmaşıklığı bulmak için sadece döngüleri saymakla yetinmeyin.
  - 2 içi içe döngünün 1 den N^2 kadar döndüğünü düşünürsek karmaşıklık $O(N^4)$ olur.
- $O(2N^2)$ veya $O(N^2+N)$ gibi ifadeler kullanmayın.
  - Sadece baskın terim kullanılır.
  - Öndeki sabitler kaldırılır.
- İç içe döngüler karmaşıklığı direk etkilerken art arda gelen döngüler karmaşıklığı etkilemez.

Bazı Matematiksel İfadeler

S(N) = 1 + 2 + 3 + 4 + … N = $$\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$$

Karelerin Toplamı: $$\sum_{i=1}^{N} i^2 = \frac{N*(N+1)*(2n+1)}{6} \approx \frac{N^3}{3}$$

Geometrik Seriler: $$\sum_{i=0}^{N} A^i = \frac{A^{N+1}-1}{A-1}$$ A > 1

$$\sum_{i=0}^{N} A^i = \frac{1-A^{N+1}}{1-A} = \Theta(1)$$ A < 1

Bazı Matematiksel İfadeler

Lineer Geometrik seriler:

$$\sum_{i=0}^{n} ix^i = x + 2x^2 + 3x^3 + \ldots + nx^n = \frac{(n-1)x^{(n+1)} - nx^n + x}{(x-1)^2}$$

Harmonik seriler:

$$H_n = \sum_{i=1}^{n} \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} = (\ln n) + O(1)$$

Logaritma:

$$\log A^B = B * \log A$$

$$\log(A * B) = \log A + \log B$$

$$\log\left(\frac{A}{B}\right) = \log A - \log B$$

Bazı Matematiksel İfadeler

- İki sınır arasındaki sayıların toplamı:

$$\sum_{i=a}^{b} f(i) = \sum_{i=0}^{b} f(i) - \sum_{i=0}^{a-1} f(i)$$

$$\sum_{i=1}^{n} (4i^2 - 6i) = 4\sum_{i=1}^{n} i^2 - 6\sum_{i=1}^{n} i$$

En iyi zaman (Tbest)

- Bir algoritma için, yürütme zamanı, maliyet veya karmaşıklık hesaplamalarında en iyi sonucun elde edildiği duruma “en iyi zaman” denir.

- Örneğin bir dizide yapılan aramanın en iyi durumu, aranan elemanın dizinin ilk elemanı olmasıdır.

En iyi zaman (Tbest)

	Birim Zaman (Unit Time)	Frekans (Frequency)	Toplam (Total)
i=1	1	1	1
while (a[i] ≠key) and (i≤ N)	2	1	2
i++	1	0	0
if (a[i]=key) print found(veya return i) else print not found(veya return 0)	2	1	2
	$T_{best}(N) = 5 = \Theta(1)$		

En kötü zaman (Tworst)

- En kötü zaman, tüm olumsuz koşulların oluşması durumunda algoritmanın çözüm üretmesi için gerekli hesaplama zamanıdır.

- Örneğin bir dizi içinde arama yapılması durumunda en kötü durum aranan elemanın dizide olmamasıdır. Çünkü aranan elemanın dizide olmadığının anlaşılması için bütün elemanlara tek tek bakılması gerekir.

En kötü zaman (Tworst)

	Birim Zaman (Unit Time)	Frekans (Frequency)	Toplam (Total)
i=1	1	1	1
while (a[i] ≠key) and (i≤ N)	2	N+1	2N+2
i++	1	N	N
if (a[i]=key) print found(veya return i) else print not found(veya return 0)	2	1	2
	$T_{worst}(N) = 3N+5 = \Theta(N)$ $T(N) \neq \Theta(N)$ $T(N) = O(N)$		

Ortalama zaman (Taverage)

- Ortalama zaman, giriş parametrelerin en iyi ve en kötü durum arasında gelmesi ile ortaya çıkan durumda harcanan zamandır.

- Bu işletim süresi, her girdi boyutundaki tüm girdilerin ortalamasıdır. n elemanın her birinin aranma olasılığının eşit olduğu varsayıldığında ve liste dışından bir eleman aranmayacağı varsayıldığında ortalama işletim süresi (n+1)/2'dir. İkinci varsayım kaldırıldığında ortalama işletim süresi [(n+1)/2,n] aralığındadır (aranan elemanların listede olma eğilimine bağlı olarak). Ortalama durum analizi basit varsayımlar yapıldığında bile zordur ve varsayımlar da gerçek performansın iyi tahmin edilememesine neden olabilir.

Ortalama zaman (Taverage)

	Birim Zaman (Unit Time)	Frekans (Frequency)	Toplam (Total)
i=1	1	1	1
while (a[i] ≠key) and (i≤ N)	2	k+1	2k+2
i++	1	k	k
if (a[i]=key) print found(veya return i) else print not found(veya return 0)	2	1	2
			=3k+5

Ortalama zaman (Taverage)

$$T_{average}(N) = \sum_{k=0}^{N-1} \frac{1}{2N}(3k+5) \quad + \quad \frac{1}{2}(3N+5)$$

$$T_{average}(N) = \frac{1}{2N}5 + \frac{1}{2N}(3+5) + \frac{1}{2N}(3.2+5) + \ldots \quad \frac{1}{2N}(3(N-1)+5) + \frac{1}{2}(3N+5)$$

$$T_{average}(N) = \frac{1}{2N}(3\frac{N(N-1)}{2} + 5N) + \frac{3}{2}N + \frac{5}{2}$$

$$T_{average}(N) = \frac{3}{4}N - \frac{3}{4} + \frac{5}{2} + \frac{3}{2}N + \frac{5}{2} \cong 2N + 4 = \theta(N)$$

Algoritma Analizinde Bazı Kurallar

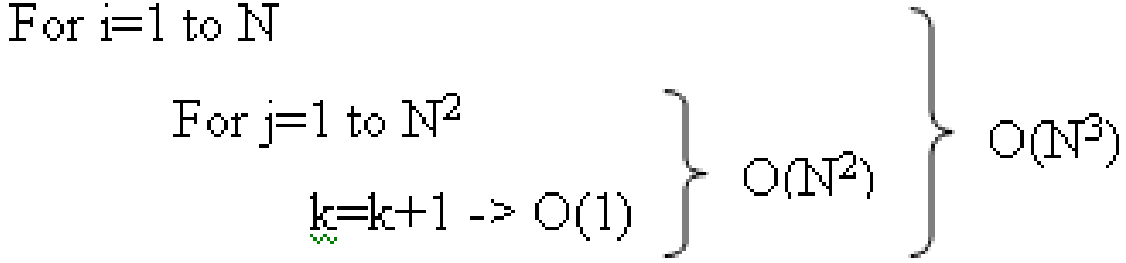
- **For Döngüsü:**
- Bir **For** döngüsü için yürütme zamanı en çok **For** döngüsünün içindeki (test dahil) deyimlerin yinelenme sayısı kadardır**.**

- **İç içe döngüler (Nested Loops)**
- İç içe döngülerde grubunun içindeki deyimin toplam yürütme zamanı, deyimlerin yürütme sürelerinin bütün **For** döngülerinin boyutlarının çarpımı kadardır. Bu durumda analiz içten dışa doğru yapılır.

$$T(N) = \sum_{i=1}^{N} \sum_{j=1}^{N} 1 = \sum_{i=1}^{N} N = N * N = N^2$$

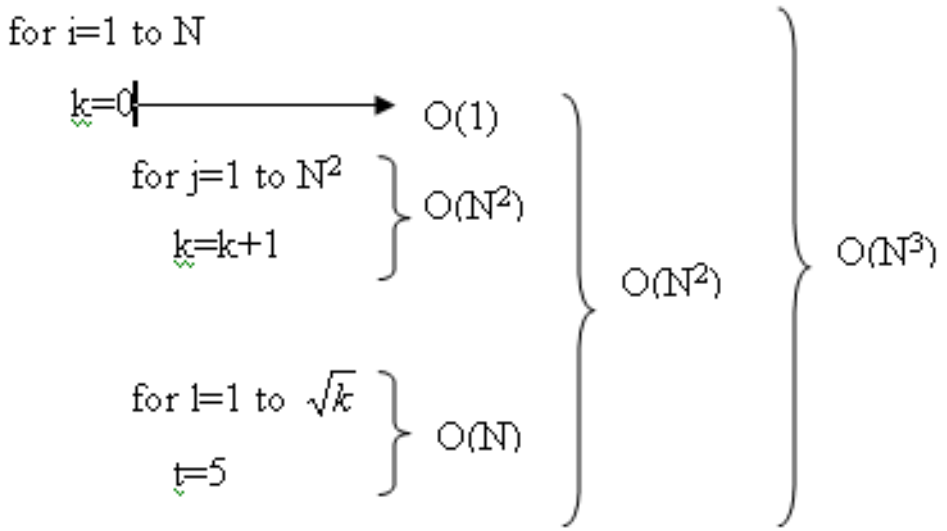
Algoritma Analizinde Bazı Kurallar

- **For Döngüsü:**



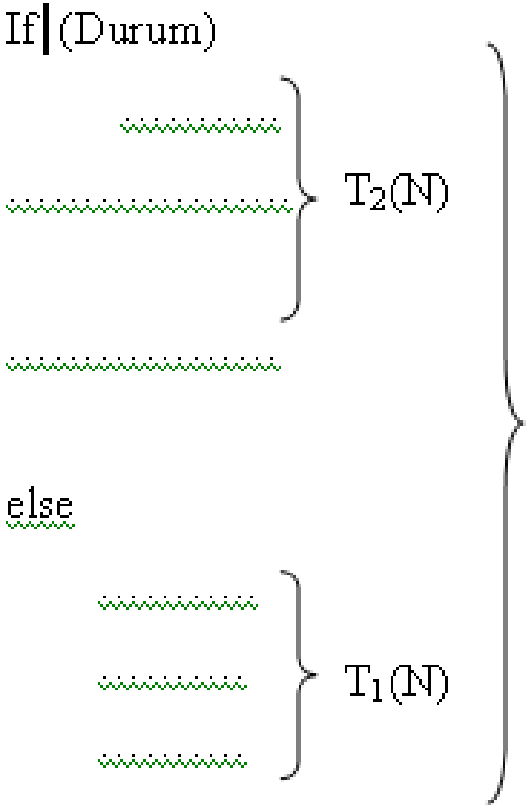
Algoritma Analizinde Bazı Kurallar

- **For Döngüsü:** Ardışık deyimlerin toplam yürütme zamanını bulabilmek için sadece toplama yapılır.



$$T_1 = O(1)$$
$$T_2 = O(N^2)$$
$$T_3 = O(N)$$
$$T_1 + T_2 + T_3 = O(N^2)$$

Algoritma Analizinde Bazı Kurallar



Durumun maksimum olduğu değerle durumu hesaplama süresinin toplamıdır.

Dolayısıyla değerlendirme zamanı $T+\max(T_1(N),T_2(N))$

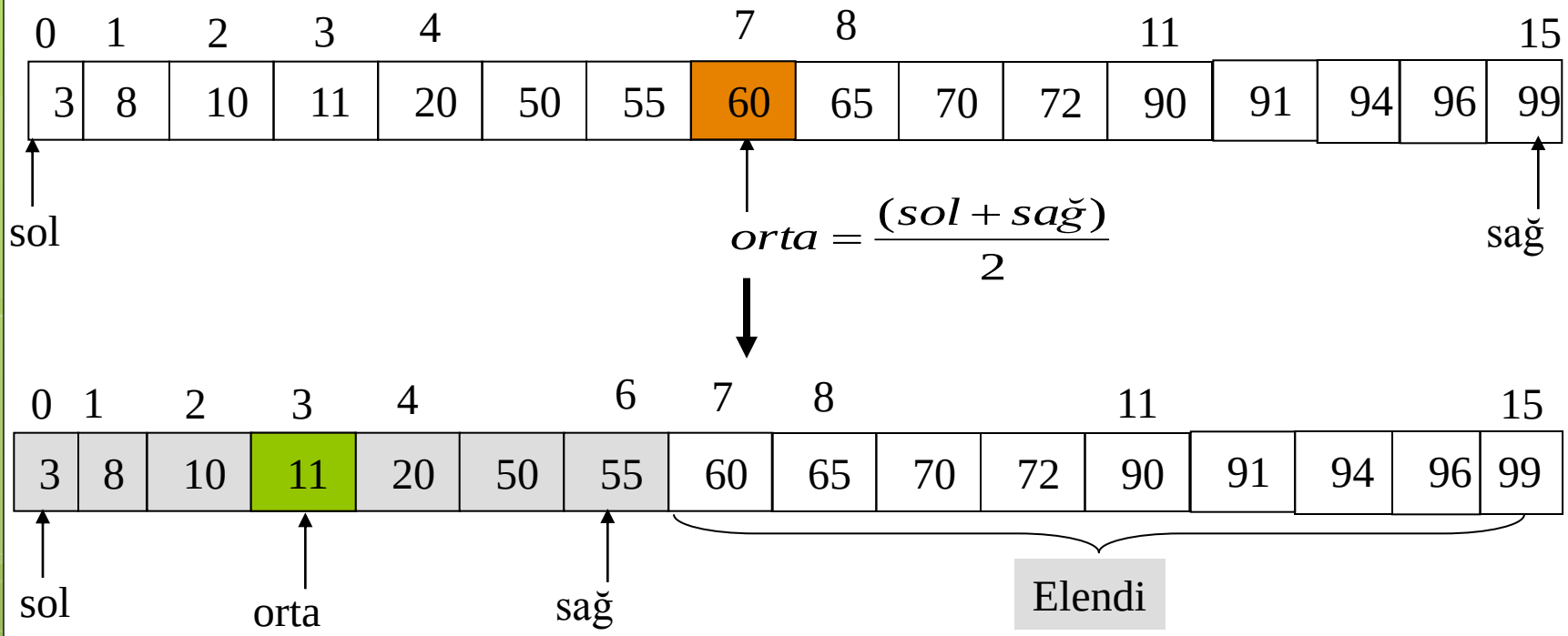
$O(\log N)+\max(O(N),O(N^2)) = O(N^2)$ olur.

Algoritma Analizinde Bir Örnek: İkili Arama Algoritması

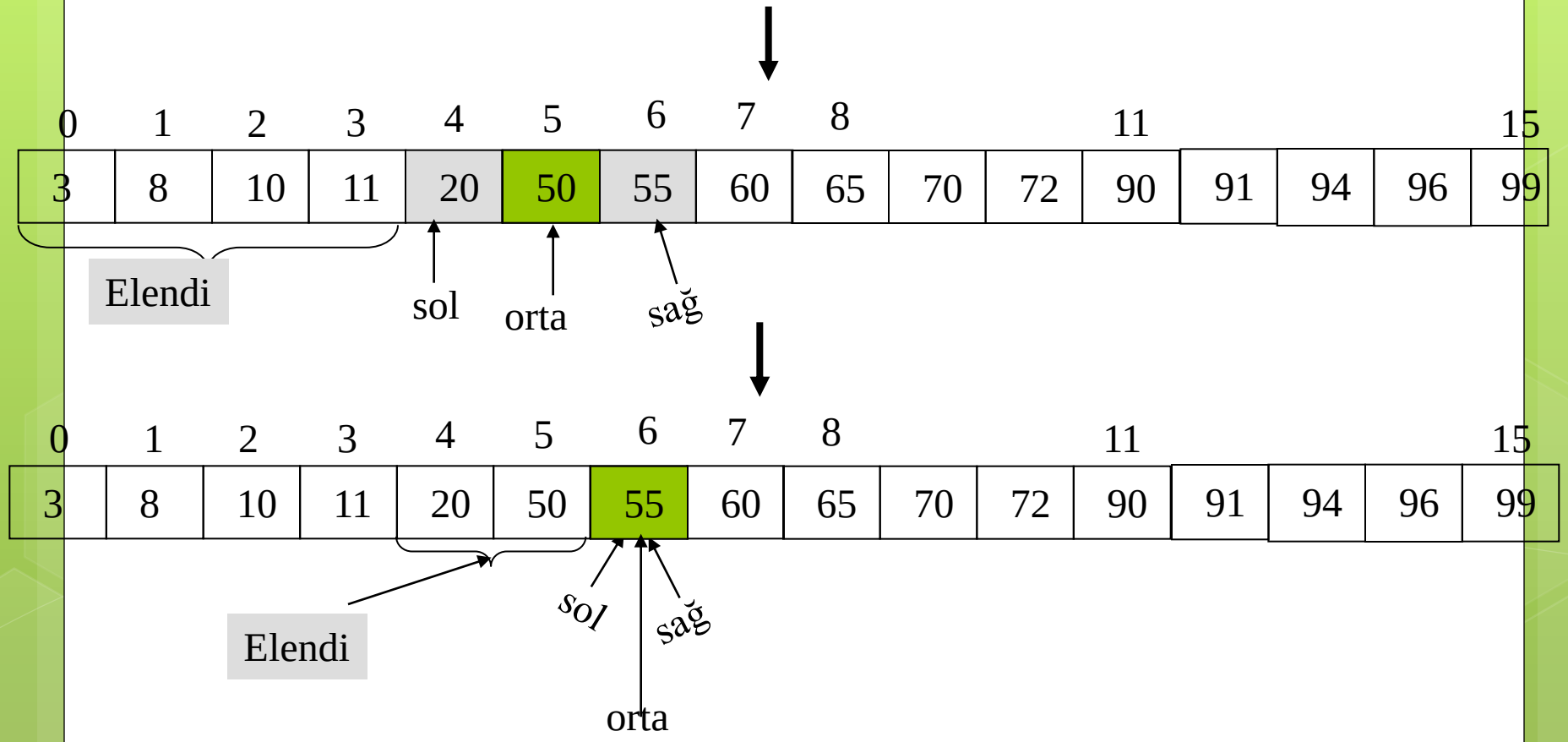
- İkili arama algoritması, sıralı dizilerde kullanılan bir arama metodur. Algoritma iteratif veya tekrarlamalı (recursive) olabilir.
- İterasyon ifadeleri (döngü) belirli bir koşul sağlanana kadar, verilen talimatların çalıştırılmasını sağlar. Belirlenen koşul, “for” döngüsündeki gibi, önceden belirlenebilir veya “while-do” döngüsünde olduğu gibi net olarak belirlenmemiş, uçu açık da olabilir.
- Aşağıda iteratif olarak tasarlanmış ikili arama algoritmasına bir örnek verilmiştir.

Örnek : İkili Arama

- Dizi sıralanmış olduğundan, dizinin ortasında bulunan sayı ile aranan sayıyı karşılaştırarak arama boyutunu yarıya düşürülür ve bu şekilde devam edilir.
- Örnek: 55'i arayalım

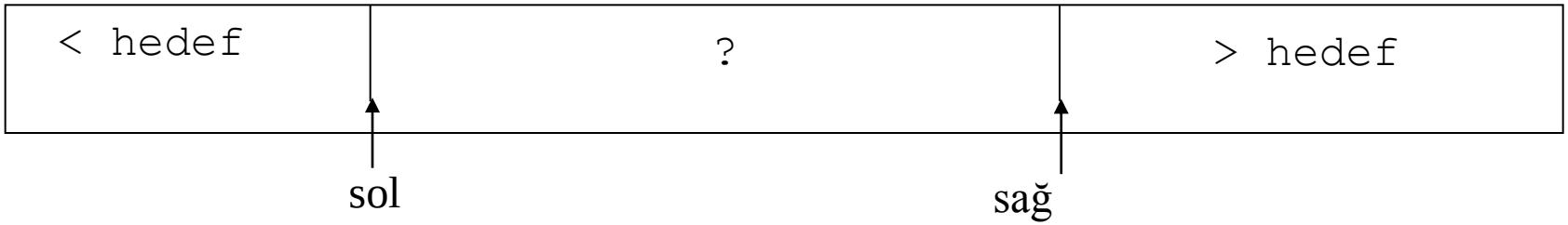


İkili arama (devam)



- 55'i bulduk→ Başarılı arama
- 57'yi aradığımızda, bir sonraki işlemde başarısız bir şekilde sonlanacak.

İkili Arama (devam)



- Hedefi ararken herhangi bir aşamada, arama alanımızı “sağ” ile “sol” arasındaki alana kısıtlamış oluyoruz.
- “sol” ’un solunda kalan alan hedeften küçüktür ve bu alan arama alanından çıkarılır.
- “sağ” ın sagında kalan alan hedeften büyüktür ve bu alan arama alanından çıkarılır.

İkili Arama - Algoritma

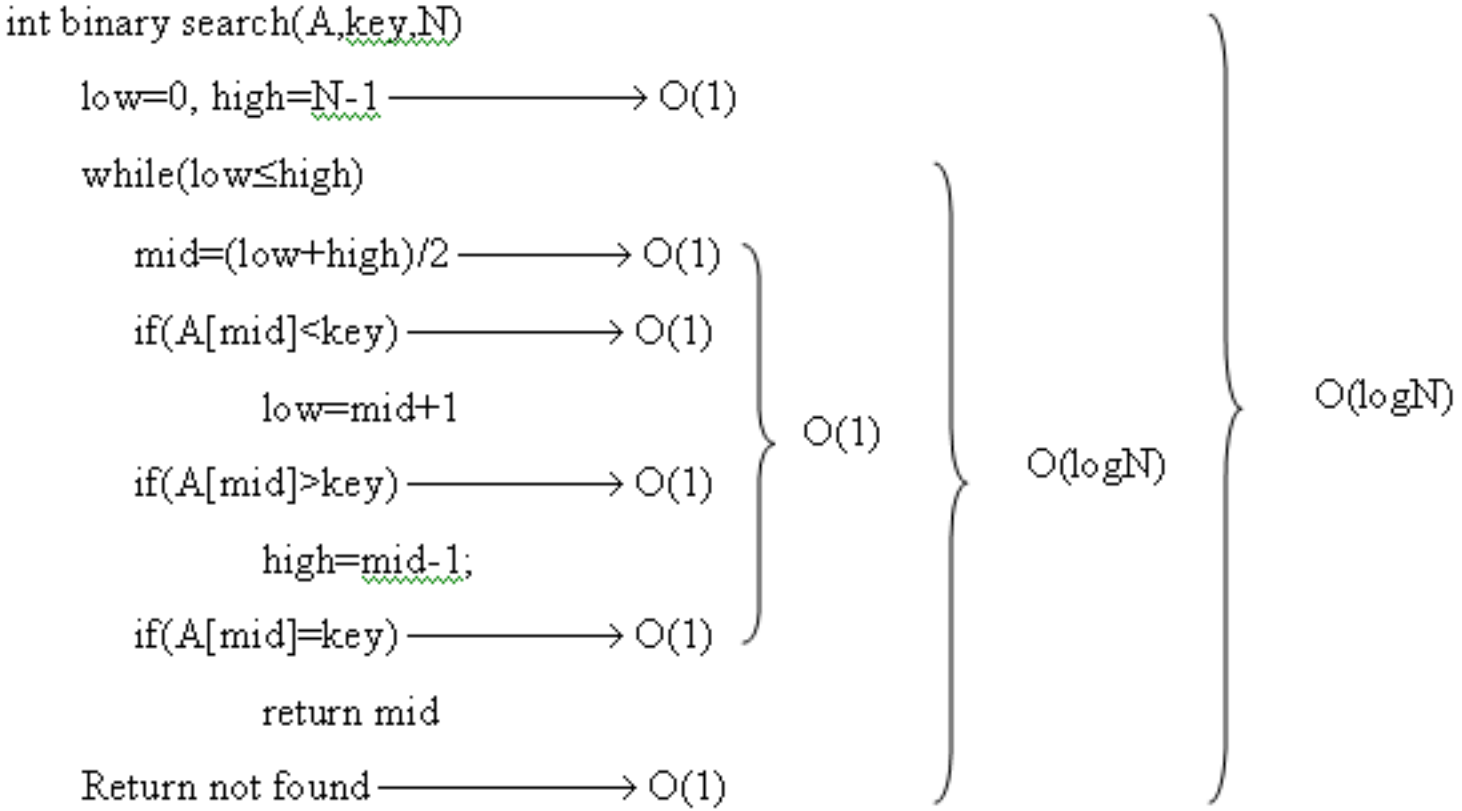
```
// Aranan sayının indeksini döndürür aranan sayı bulunamazsa -1 döndürür.
int ikiliArama(int A[], int N, int sayi){
    sol = 0;
    sag = N-1;

    while (sol <= sag){
        int orta = (sol+sag)/2;              // Test edilecek sayının indeksi
        if (A[orta] == sayi) return orta;  // Aranan sayı bulundu. İndeksi döndür
        else if (sayi < A[orta]) sag = orta – 1;       // Sağ tarafı ele
        else sol = orta+1;                                  // Sol tarafı ele
    } //bitti-while

    return –1;   // Aranan sayı bulunamadı
} //bitti-ikiliArama
```

- En kötü çalışma zamanı: $T(n) = 3 + 5*\log_2 N$. Neden?

Algoritma Analizinde Bir Örnek: İkili Arama Algoritması



Algoritma Analizinde Bir Örnek: İkili Arama Algoritması

- İlk çalışmada N eleman var.
- 2. de $\frac{N}{2^1} \rightarrow 3.de \frac{N}{2^3} \rightarrow \ldots \frac{N}{2^k} = 1 \rightarrow 0$
- Yukarıdaki ifadeden $N=2^k$ olduğu anlaşılmaktadır. Bu durumda k=logN olarak belirlenir. N=1 için k değeri 0 olacağından,
- Tworst =logN+1 olacaktır.

Ödev

- Diğer notasyonlar hakkında araştırma yapıp, ders notundaki örneklere uygulayınız.
- Farklı algoritmalara tüm notasyonları uygulayıp (T(n) hesabı) sonuçları yorumlayınız.(En az 5 algoritma)
- Ödevler gelecek hafta Word dokümanı, slayt ve çıktı olarak gelecektir. Süre dışında gelen ödevler kabul edilmeyecektir.
- Ödev raporunda hazırlayan kişi, çalışma süresi, bulduğu kaynaklar ve katkısı yazılacaktır.
- Ödev teslimi imza ile alınacaktır.

Örnekler

Örnek I:
Dizideki sayıların toplamını bulma

```
int Topla(int A[], int N)
{
  int toplam = 0;

  for  (i=0; i < N; i++){
    toplam += A[i];
  } //Bitti-for

  return toplam;
} //Bitti-Topla
```

- Bu fonksiyonun yürütme zamanı ne kadardır?

Örnek I:
Dizideki sayıların toplamını bulma

```
int Topla(int A[], int N)
{
  int topla = 0;
  for (i=0; i < N; i++){
    topla += A[i];
  } //Bitti-for

  return topla;
} //Bitti-Topla
```

İşlem sayısı

1

N

N

1

Toplam: 1 + N + N + 1 = 2N + 2

- Çalışma zamanı: T(N) = 2N+2
  - N dizideki sayı sayısı

Örnek II:
Dizideki bir elemanın aranması

```
int Arama(int A[], int N,
                  int sayi) {
  int i = 0;
  while (i < N){
    if (A[i] == sayi) break;
    i++;
  } //bitti-while

  if (i < N) return i;
  else return -1;
} //bitti-Arama
```

İşlem sayısı

1

1<=L<=N

1<=L<=N

0<=L<=N

1

1

Toplam: 1+3*L+1+1 = 3L+3

Örnek II:
Dizideki bir elemanın aranması

- En iyi çalışma zamanı nedir?
  - Döngü sadece bir kez çalıştı=>T(n) = 6
- Ortalama(beklenen) çalışma zamanı nedir?
  - Döngü N/2 kez çalıştı =>T(n)=3*n/2+3 = 1.5n+3
- En kötü çalışma zamanı nedir?
  - Döngü N kez çalıştı =>T(n) = 3n+3

Örnek III: Matris Çarpımı

```
/* İki boyutlu dizi A, B, C.      Hesapla C = A*B    */
for (i=0; i<N; i++) {
    for (j=0; j<N; j++) {
        C[i][j] = 0;
        for (int k=0; k<N; k++){
          C[i][j] += A[i][k]*B[k][j];
        } //bitti-en içteki for
    } //bitti-içteki for
} //bitti-dıştaki for
```

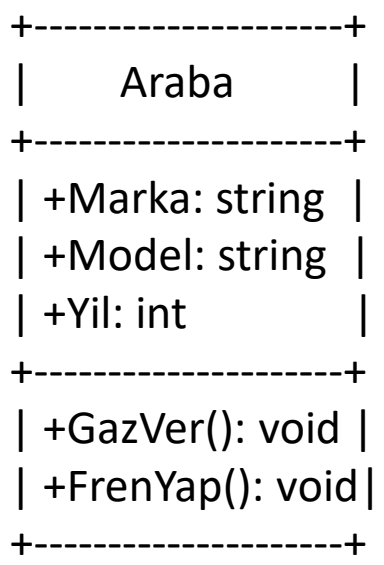
$$T(N) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (1 + \sum_{k=0}^{N-1} 1) = N^3 + N^2$$

OOP

- **Sınıf (Class):**

Bir sınıf, nesnelerin (objelerin) temel taslağını ve tanımını sağlayan bir şablondur. Sınıflar, bir nesnenin özelliklerini (veri elemanları veya özellikleri) ve bu nesnenin yapabileceği işlevleri (metodları) tanımlar.

```
public class Araba
{
    // Özellikler (veri elemanları)
    public string Marka;
    public string Model;
    public int Yil;
    // Metotlar
    public void GazVer()
    {
        // Gaz verme işlemleri
    }
    public void FrenYap()
    {
        // Fren yapma işlemleri
    }
}
```

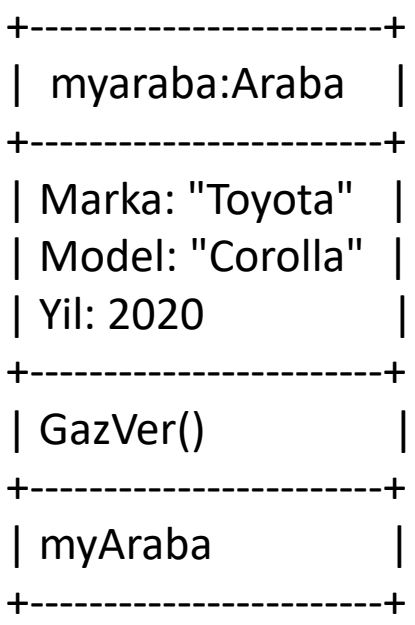


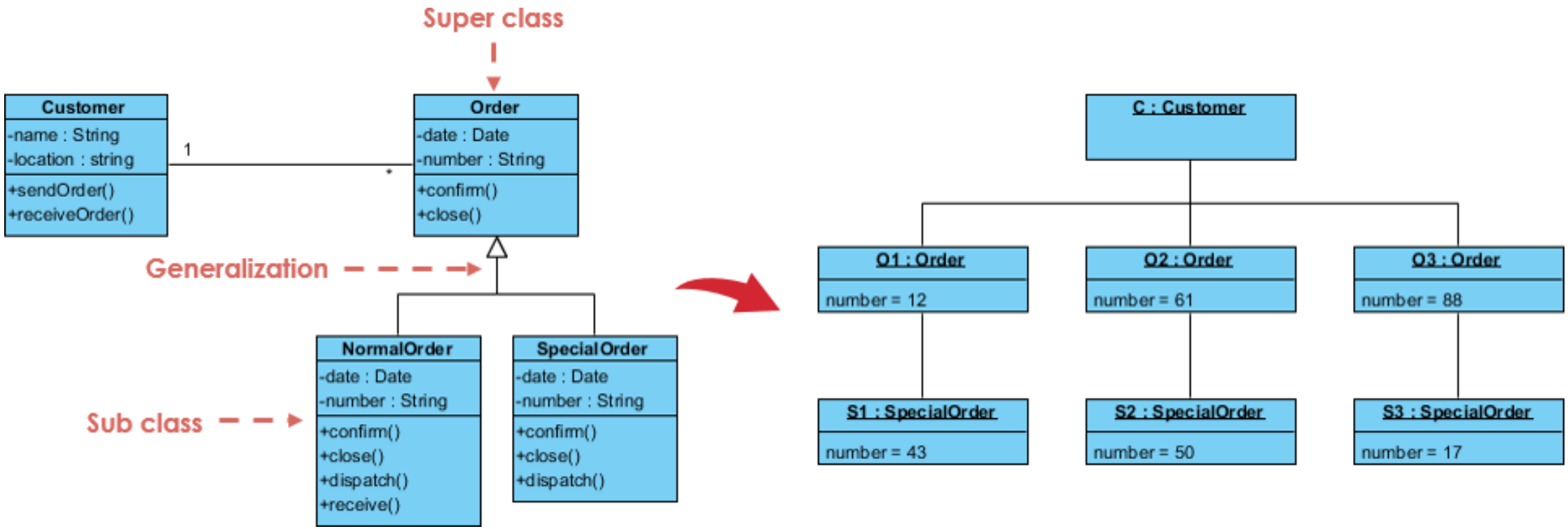
- **Nesne (Object):**

Nesne, bir sınıftan türetilen somut bir örnek veya örneklerdir. Bir sınıf, bir nesne oluşturulduğunda, bellekte o nesnenin özelliklerini ve işlevselliğini barındıran bir örneği oluşturulmuş olur.

```
Araba myAraba = new Araba();
myAraba.Marka = "Toyota";
myAraba.Model = "Corolla";
myAraba.Yil = 2020;

myAraba.GazVer(); // Araba sınıfındaki GazVer metodu çağrılıyor.
```




Super class
Customer
-name : String
-location : string
+sendOrder()
+receiveOrder()
1
*
Order
-date : Date
-number : String
+confirm()
+close()
Generalization
NormalOrder
-date : Date
-number : String
+confirm()
+close()
+dispatch()
+receive()
SpecialOrder
-date : Date
-number : String
+confirm()
+close()
+dispatch()
Sub class
C : Customer
O1 : Order
number = 12
O2 : Order
number = 61
O3 : Order
number = 88
S1 : SpecialOrder
number = 43
S2 : SpecialOrder
number = 50
S3 : SpecialOrder
number = 17

Polymorphism

```
int topla(int x, int y) {
    return x + y;
}

double topla(double x, double y) {
    return x + y;
}

int main() {
    int sonuc1 = topla(3, 4);       // topla(int, int) çağrılır
    double sonuc2 = topla(2.5, 3.7); // topla(double, double) çağrılır
    return 0;
}
```

```
// Üst sınıf (Base Class)
class GeometrikSekil {
    protected double genislik;
    protected double yukseklik;

    // Parametre almayan constructor
    public GeometrikSekil() {
        genislik = 0;
        yukseklik = 0;
    }
    // Parametreli constructor
    public GeometrikSekil(double genislik, double yukseklik) {
        this.genislik = genislik;
        this.yukseklik = yukseklik;
    }
    public void ciz() {
        System.out.println("Bir geometrik şekil çizildi.");
    }
}
// Alt sınıf (Derived Class)
class Dikdortgen extends GeometrikSekil {
    // Alt sınıfın constructor'ı, üst sınıfın constructor'ını çağırıyor
    public Dikdortgen(double genislik, double yukseklik) {
        super(genislik, yukseklik);
    }
    // Üst sınıftan miras alınan metodu override ediyor
    public void ciz() {
        System.out.println("Dikdörtgen çizildi. Genişlik: " + genislik
+ ", Yükseklik: " + yukseklik);
    }
}

public class Main {
    public static void main(String[] args) {
        // Alt sınıf nesnesi oluşturuluyor
        Dikdortgen dikdortgen = new Dikdortgen(5, 3);

        // Alt sınıfın metodu çağrılıyor, ancak override edildiği için
alt sınıfın davranışı gösteriliyor
        dikdortgen.ciz();

        // Üst sınıf referansıyla alt sınıf nesnesine erişim
        GeometrikSekil sekil = dikdortgen;
        sekil.ciz(); // Alt sınıfın metodu çağrılır, çünkü override
kullanıldı
    }
}
```

İŞLETİM SİSTEMLERİ

(Operating Systems)

İşletim Sistemi Tanımı, Görevleri, Bilinen İşletim Sistemleri

Çok Kullanıcılı Sistemler, Bellek Yönetim Birimi

Linux ve Windows Ailesi, Bilinen İşletim Sistemleri



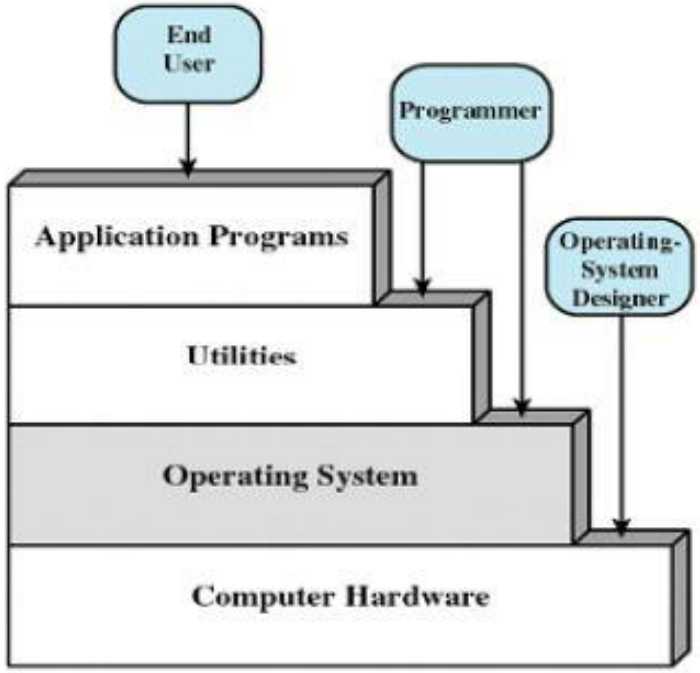
muratcanguzelay.blogspot.com.tr

İşletim Sistemi- Operating System- Nedir?

- Uygulama yazılımları ile bilgisayar donanımı arasındaki bağlantıyı sağlayan özel yazılımlardır.
- Bilgisayarda programların çalıştırılmasını ve istenilen temel işlevlerin yerine getirilmesini işletim sistemi sağlar.
- Dosya ve dizin oluşturma, silme, kopyalama, taşıma işlemleri, Disk biçimlendirilmesi (format), tarih, saat, donanım ayarları, vb. işlemler işletim sistemi tarafından gerçekleştirilir.
- Bilgisayarın çalışması için işletim sisteminin bilgisayara yüklü olması gerekir; işletim sistemi olmadan bilgisayar çalışmaz.

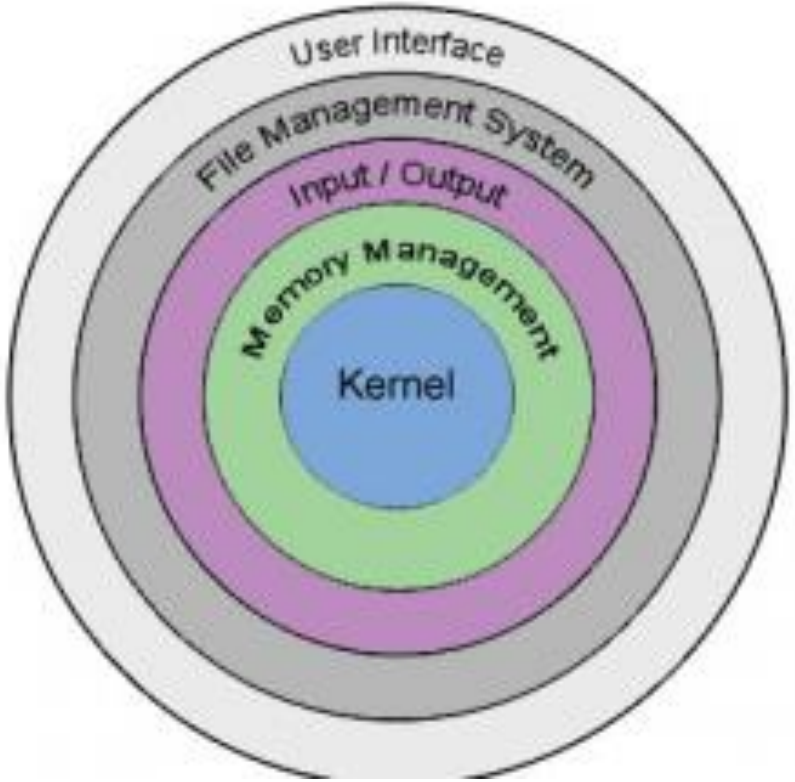
İşletim Sistemi- Bilgisayar Katmanlarındaki Yeri



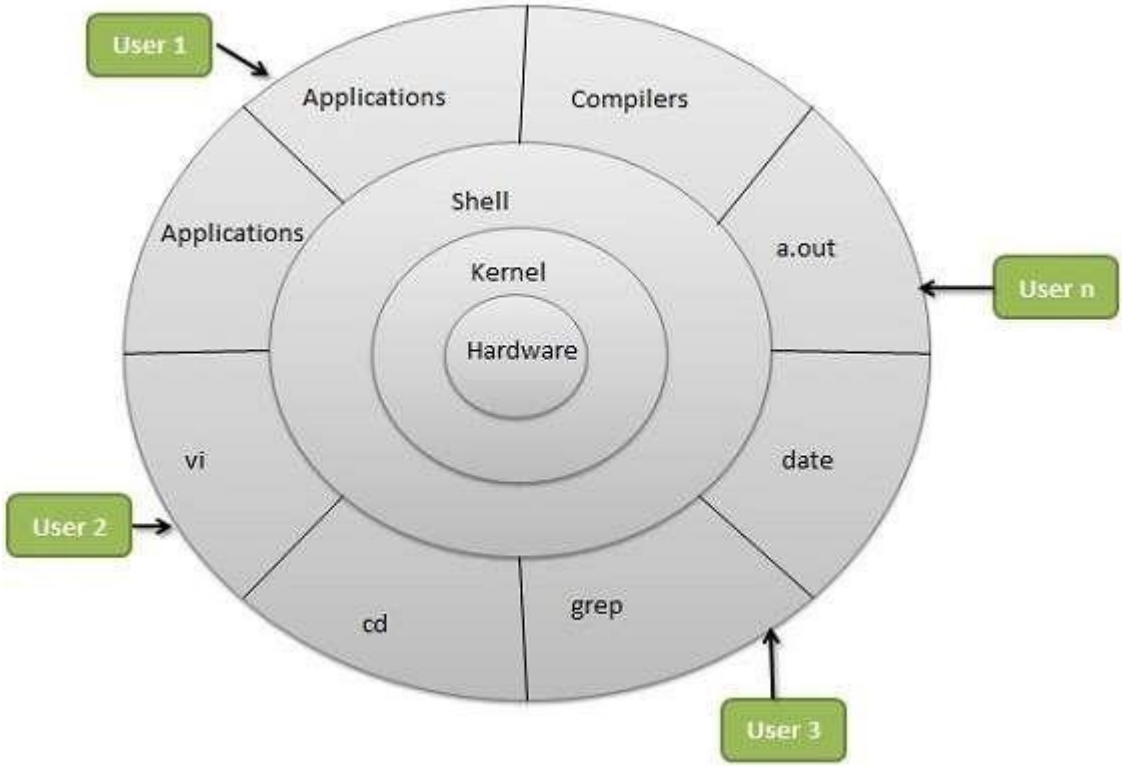


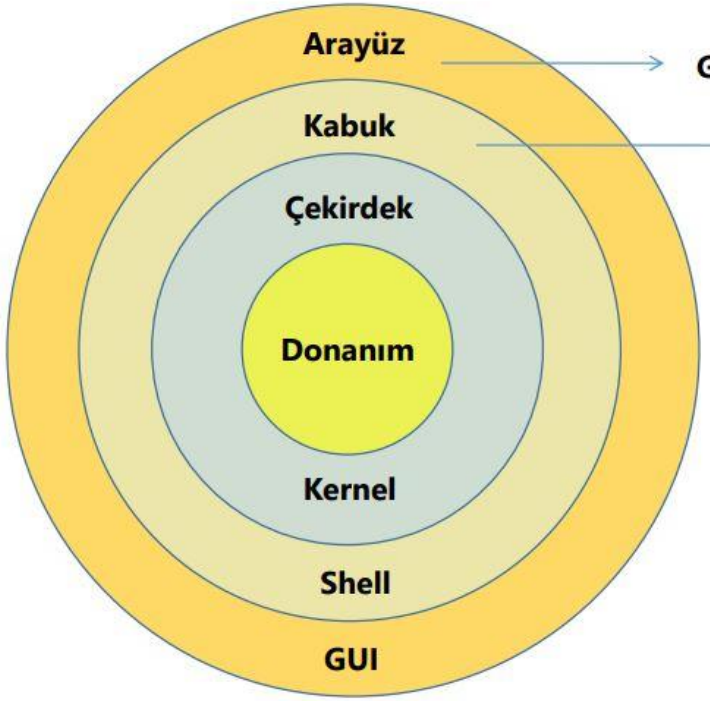
Operating Systems Layer- İşletim Sistem katmanları

- **User Interface – Kullanıcı arayüzü**
- **File Management System- Dosya yönetimi**
- **Input / Output- Giriş Çıkış**
- **Memory Management- Hafıza yönetimi**
- **Kernel-çekirdek**



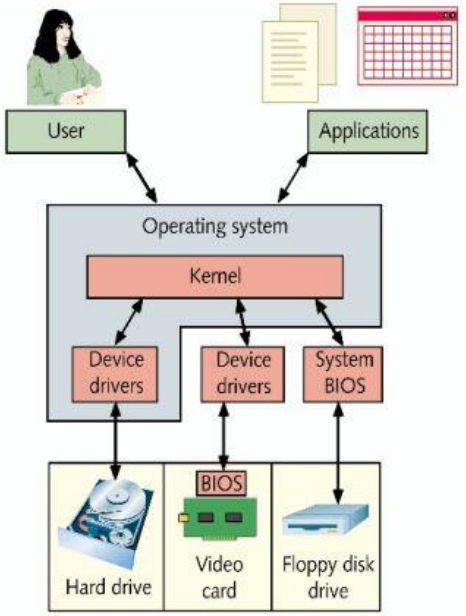
İşletim Sistemi Genel Yapısı



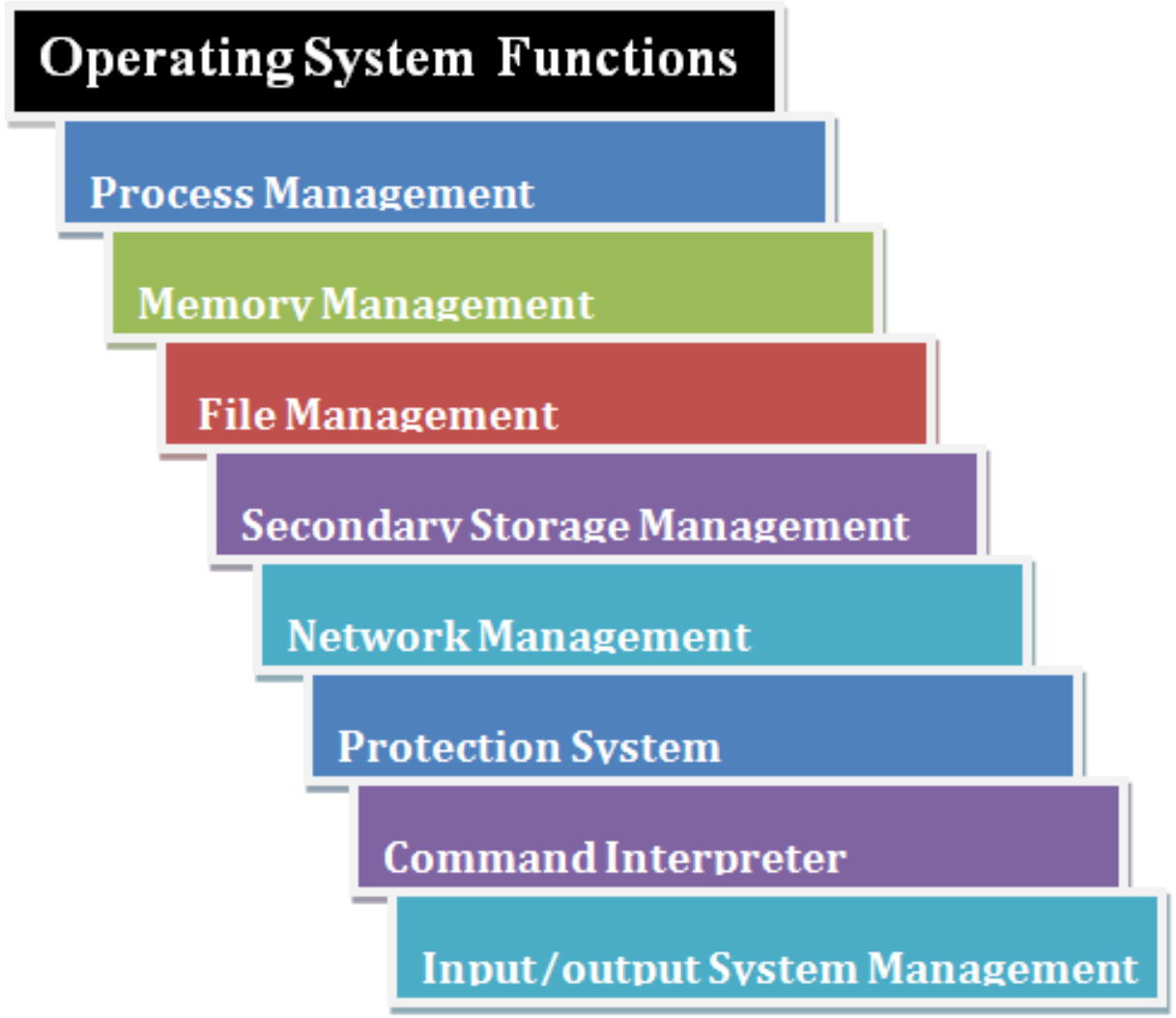


Grafiksel ve Komut Satırı Arabirimi

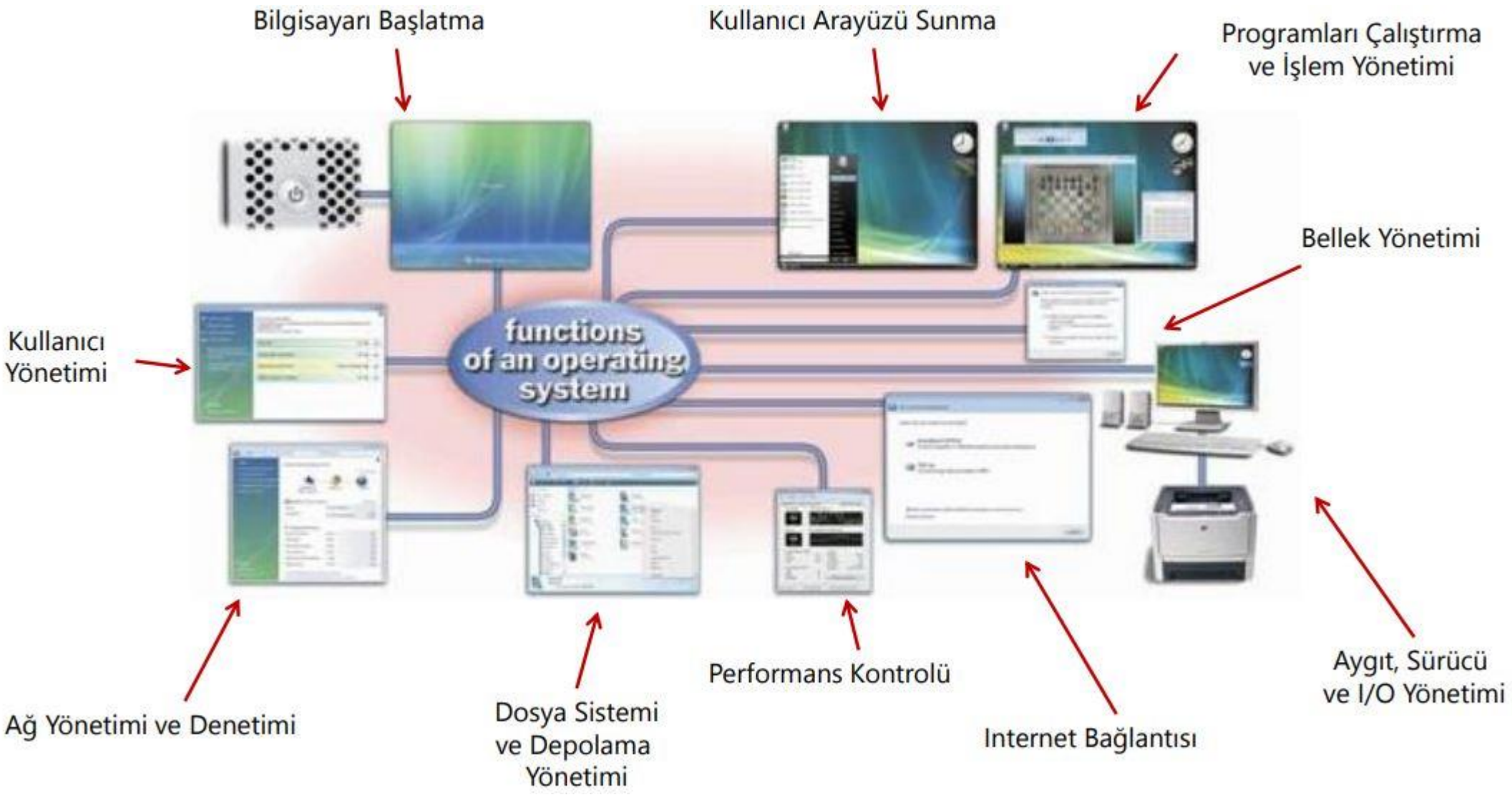
API'ler ve Komutlar



İşletim Sistemleri- Ne Yapar?



İşletim Sistemleri- Ne Yapar?



Kernel Nedir? Ne iş yapar?

- Çekirdek olarak adlandırılır, İşletim sisteminin kalbidir
- Uygulamalar ve donanım seviyesindeki bilgi işlemleri arasında bir köprü görevi görür.
- Donanıma kullanıcıların ve uygulama yazılımlarının doğrudan erişimlerini sınırlandırır ve düzenler.
- Bir çok sistemde kabuk ve çekirdek ayrımı sadece kavramsaldır
- Çekirdeğin görevleri sistemin kaynaklarını yönetmeyi de kapsamaktadır.
- Giriş çıkış işlemlerinin yönetilmesi (I/O management), İşlem yönetimi (process management), Hafıza yönetimi (memory management), Aygıt yönetimi, Dosya yönetimi.

Shell Nedir? Ne işe Yarar?

- Kabuk katmanı çekirdeğe kullanıcı veya uygulama arabirimlerinden gelen komutları iletmekten sorumludur.
- Aynı zamanda çekirdekten aldığı bildirimleri ve çekirdeğin geçerli durumunu arayüze iletir.
- Uygulama yazılımları, API (Application Programming Interface) adı verilen arabirimler ile kabuk katmanıyla konuşur.
- Bu anlamda API'ler de kabuk katmanına dahil edilebilirler.
- API'ler yazılım geliştiriciler için SDK olarak bilinen geliştirme kılavuzları sunarlar

Arayüz Nedir?

- Metin tabanlı arayüzler
- Grafiksel kullanıcı arayüzü



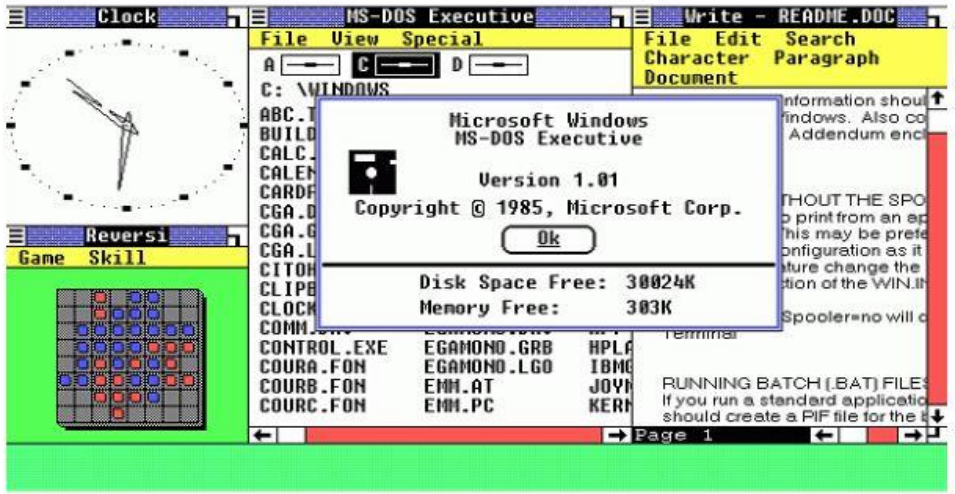

```
bash-3.2$ sdcv linus ; espeak linus
Found 2 items, similar to linus.
-->Jargon File
-->Linus

Linus /leen'us'/ or /lin'us'/, not /li:'nus/ Linus Torvalds, the author
   of {Linux}. Nobody in the hacker culture has been as readily recognized
   by first name alone since Ken (Thompson).

-->GNU/Linux English-English Dictionary
-->Linus

/leen'us'/ or /lin'us'/, not /li:'nus/ Linus Torvalds, the author of Linux. Nobody in the hacker culture has been a
s readily recognized by first name alone since Ken (Thompson). From Jargon Dictionary http://www.tldp.org/LDP/Linux
-Dictionary/

bash-3.2$ sdcv -l
Dictionary's name   Word count
Jargon File    2367
The Devil's Dictionary    986
GNU/Linux English-English Dictionary    16694
bash-3.2$ sdcv
Enter word or phrase:
Enter word or phrase:
bash-3.2$
```



Bilinen İşletim Sistemleri- Linux Ailesi

- Ubuntu Linux
- Linux Mint
- Arch Linux
- Deepin
- Fedora
- Debian
- openSUSE
- Red Hat Enterprise Linux
- Ubuntu Server
- CentOS
- SUSE Enterprise Linux
- Pardus
- Gelecek
- Android!

Bilinen İşletim Sistemleri- Windows Ailesi

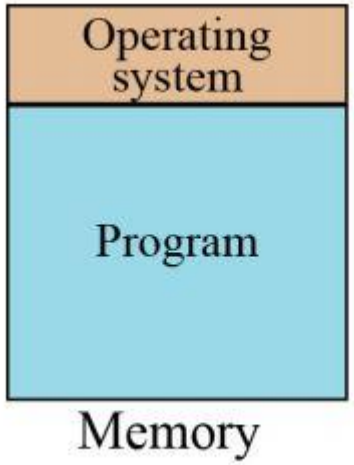
- Windows 1.0 | 20 Kasım 1985
- Windows 2.0 | Kasım 1987
- Windows 3.0 | 22 Mayıs 1990
- Windows 95 | 24 Ağustos 1995
- Windows 98 | 25 Haziran 1998
- Windows ME | 24 Eylül 2000
- Windows XP | 25 Ekim 2001
- Windows Vista | 30 Ocak 2007
- Windows 7 | 25 Ekim 2009
- Windows 8 | 26 Ekim 2012
- Windows 8.1 | 26 Ekim 2012
- Windows 10 | 1 Ekim 2014

İşletim Sistemi Türleri

- 3 sınıflama kriteri;
  - Kullanıcılara sağladıkları çalışma ortamı
  - Kullanıcıların sisteme erişim biçimleri
  - Tasarım ve mimarisinde izlenen yaklaşımlara göre
- Tekli Programlama (Monoprogramming)
- Çoklu Programlama (Multiprogramming)
- Çok görevli işlem (Multitasking)

monoprogramming

- Birim zamanda/aynı anda yalnızca bir görüntü ortam kurulur.
- Kullanıcı sistemin tüm kaynaklarını kullanabilir.
- Kaynak atama, bütünlük koruma vb. sorunlar kolaydır.

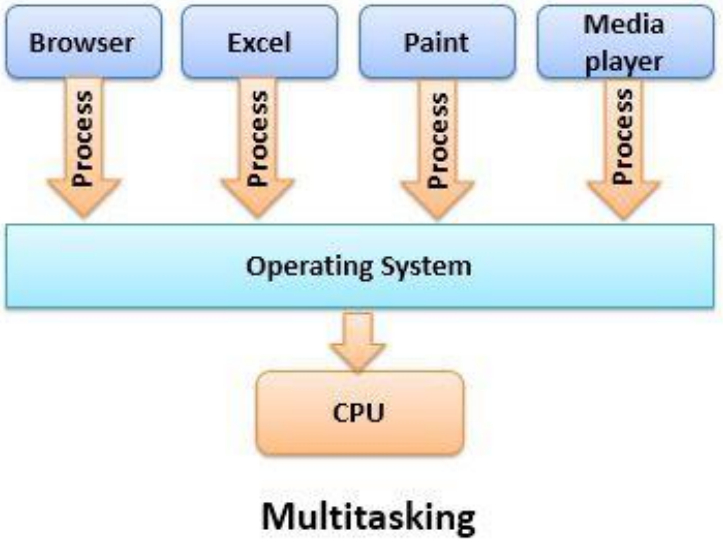


multiprogramming



- Sistemde çalışan herhangi bir iş bekleme durumuna geçtiğinde işlemcinin başka bir işe başlaması,
- İşlemci hızı >> I/O,
- Verimlilik arttırır,

- Multitasking: Birçok görev için atanmış birçok adımın birlikte çalıştırılması.



Bazı görevler

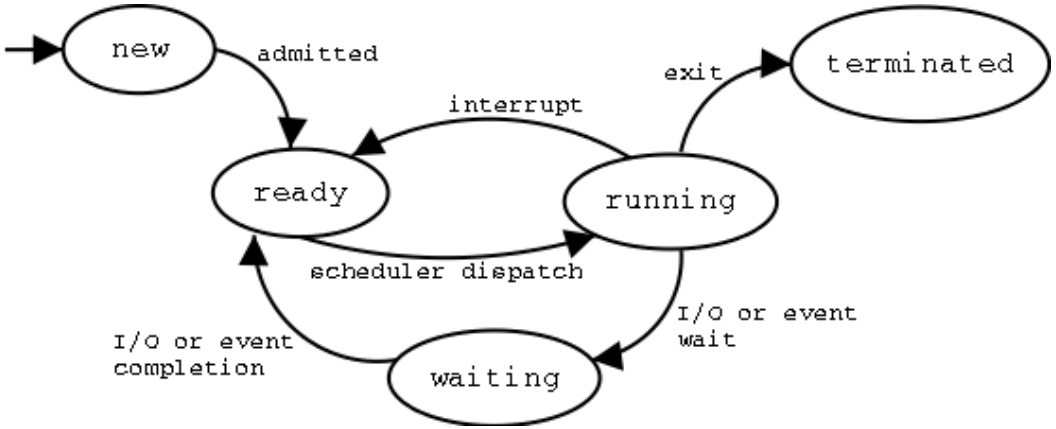
- Kesme Yönetimi: Kesme yazılım veya donanımdan kaynaklanan bir uyarı sonucunda işlemcinin yürüttüğü görevi bırakıp, ilgili uyarıyı üreten birime hizmet verecek göreve anahtarlanmasına yol açan olaydır.
- Giriş/Çıkış Yönetim Sistemi: çekirdek sistemin sağladığı veri aktarım işlevi ile daha üst katmanların giriş/çıkış istemlerini ilişkilendiren arabirim.
- Dosya Yönetim Sistemi: daha üst düzeydeki yazılım katmanlarının simgesel olarak tanımlanan dosyalar ile sistemdeki fiziksel G/Ç birimleri arasındaki eşleşmeyi yapan kesimdir.

Bazı görevler

- Proses yönetimi:
  - Çalışıyor-running
  - Askıda-waiting-blocked
  - Hazır-ready

Zaman Çizelgeleyici: Time Schedular:işlemciye atanacak prosesi ve prosesin hangi koşullar altında işlemciyi kullanacağını belirler.

  - Proses kuyruğundan yürütülecek prosesi seçmek
  - Prosese ait kullanım zaman dilimini ayarlamak



Bazı Görevler

- Deadlock- Ölümcül Kilitlenme

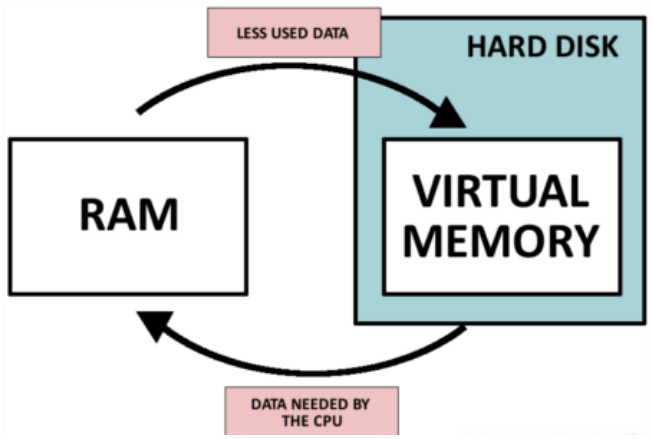
Proseslerin hiçbir zaman ele geçiremeyecekleri bir birime ya da kaynağa ihtiyaç duymaları durumunda sürekli olarak askıda kalmalarıdır.

Bir prosesin işinin bitmesi bir başka prosesin işinin bitmesini bekleyen bir durum oluşturuyorsa ölümcül kilitlenme oluşur.

Semafor: genel olarak kaynakların kritik bölgelerde kilitlenme olmaması için kullanılan bir değişken türüdür. Üzerinde iki temel işlem tanımlıdır. Wait (s) semaforun içeriğini bir azaltır. Signal (s) semafor içeriğini bir arttırır.

Bellek yönetimi

- İhtiyaç duyulan belleği vermek, bununla ilgili bilgileri tutmak, kullanılmayan bellek alanını serbest bırakmak, sanal bellek yönetimi mekanizması oluşturmak ve belleğin verimli kullanılmasını sağlamak temel işlevleridir.
- Sanal Bellek:



Kaynaklar

- Hitit Üni. Mustafa Coşar, İşletim Sistemi.

PROGRAMLAMA DİLLERİ

- Program
- Yazılım Geliştirme Süreci
- Programlama Dilleri Tarihçesi
- Kod Sözdizimi
- Nesneye Yönelik Programlama
- Tür Kontrolü
- Alt Programlar
- Programlama Dilleri
- Programlama Dillerinin Önemi
- Dilleri Sınıflandırılması
- Anlambilim
- BNF Notasyonu
- Kontrol Deyimleri

Muhammet Baykara
mbaykara@firat.edu.tr

Program Nedir?

- **Program:** Bilgisayarın bir işlevi yapması için tasarlanmış komutlar zinciridir.
- Ardışık simgeler dizisi.
- **Programlama Dili:** Bir makinenin davranışını kontrol etmek için kullanılabilen ve program üretimi için kullanılan yapay dil.



Yazılım Geliştirme Süreci

- **Gereksinim Analizi:** Yazılım projesinin ilk etabında projenin ihtiyaç duyduğu ana modüller analiz edilmeli, (software requirements specification) proje amaçları ve hedefleri detaylandırılmalıdır.
- **Tasarım:** Oluşturmak istenilen proje web tabanlı, mobil veya masaüstü olabilir. Bu doğrultuda yapacağınız tasarımın bu platformlara veya cihazlara uygun olması gerekmektedir.
- **Kodlama:** Güçlü bir yazılım mimarisi ile çalışılmalı ve sonradan çıkabilecek tüm isteklere kolaylıkla cevap verebilecek şeklide kodlama yapılmalıdır.



Yazılım Geliştirme Süreci

- **Sertifikasyon (Test):** Önceden belirlenen gereksinimlerin karşılanıp, karşılanmadığı doğru çıktıyı üretip, üretmediği testleri yapılmalıdır. Kalite denetimi yapılarak kullanıcıya sunulmasıdır.
- **Bakım:** Teslimat sonrasında çıkan sorunları gidermek ve performansı artırmak için yazılım uygulamasını değiştirmek ve güncellemektir. Bu değişimler basit kodlama hatası şeklinde olabileceği gibi tasarımdan kaynaklanan hataların giderilmesi şeklinde de olabilir.

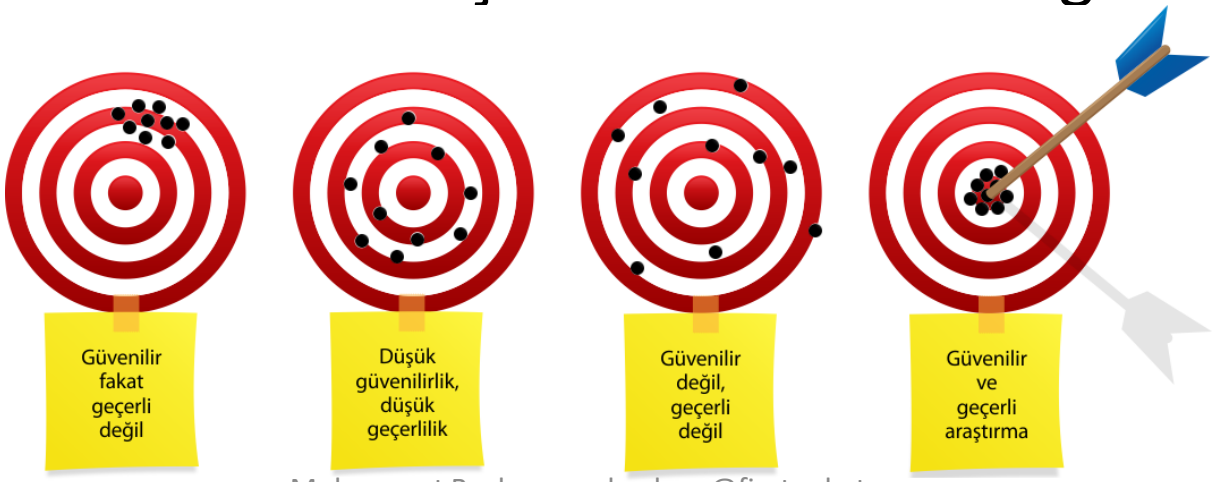


Programlama Dilinin Önemi

- **1. Yazılım Güvenilir Olmalıdır**
- Güvenli yazılımlar kullanmak programcılar için önemlidir. Kullanılan programların güvenli sayılabilmesi için bazı kalite ölçütlerine sahip olması gerekir. Bu ölçütler;
- **a. Yazılabilirlik:** Problemin gerektirdiği veri giriş çıkış, akış denetimi gibi unsurların programlama dili tarafından karşılanabilir olması ile ilgili bir kavramdır.
- **b. Okunabilirlik:** Okunabilirlik programın mantığını takip etmek, sınamak ve hata ayıklayabilmek ile ilgili bir kavramdır.
- **c. Sıra dışı durumları karşılayabilme:** Programlama dili hatalı giriş, aritmetik taşma gibi durumlara hazırlıklı olmalı bu durumlara uygun çözümleri içermelidir.

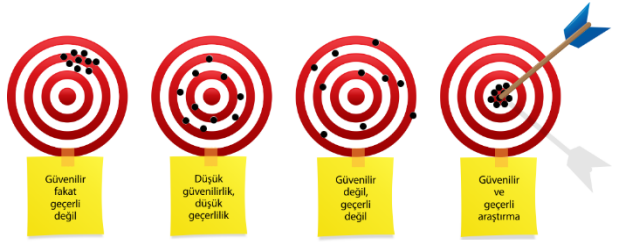
Programlama Dilinin Önemi

- **2. Yazılım Bakıma Elverişli Olmalıdır**
- Daha önceden geliştirilmiş bir yazılımın hatalarının ayıklanması ve giderilmesi, yeni modüller eklenmesi ve çıkarılması kullanıcı ihtiyaçları doğrultusunda yazılımın geliştirilmesi yazılım bakımının temel unsurlarıdır.
- Yazılımın okunabilirliğinin yanında değiştirilebilirliğinin de yüksek olması o programlama dilinin bakıma elverişli olması anlamına gelmektedir.



Programlama Dilinin Önemi

- **3. Yazılım Verimli Çalışmalıdır.**
- Geliştirilen yazılımların verimliliği kullanılan programlama dilline doğrudan bağlıdır. Verimlilik esas olarak programlama dilinin tasarımı sırasında ele alınan bir konudur.
- İlk dönem programlama dillerinde verimlilik programlama dilinin mevcut donanımı etkin ve hızlı bir şekilde kullanması ile ölçülmekteydi.
- Bu gün bu anlayış değişmiş ve programlama dilinin verimliliği işletim hızı, bellek gereksinimi, geliştirme ve bakım sırasında harcanacak zaman gibi kavramlarla ölçülen bir değer olmuştur.



Okunabilirlik Temel Teorem

- ***Kod, başkaları tarafından en az zamanda anlaşılsın diye yazılmalıdır.***

- ***Az kod her zaman daha iyi midir?***

Genellikle, daha az kod yazmak daha iyidir ama her zaman az olan iyi olacak diye bir şey yok.

```
1   // ilki
2   assert((!(bucket = FindBucket(key))) || !bucket->IsOccupied());
3
4   // ikincisi
5   bucket = FindBucket(key);
6   if (bucket != NULL) assert(!bucket->IsOccupied());
```

İyi kaynak kodu göze kolay gelmelidir! Estetik

```
// ilki
class StatsKeeper {
public:
// A class for keeping track of a series of doubles
void Add(double d); // and methods for quick statistics about them
private: int count; /* how many so far
*/ public:
      double Average();
private: double minimum;
list<double>
  past_items
  ;double maximum;
};

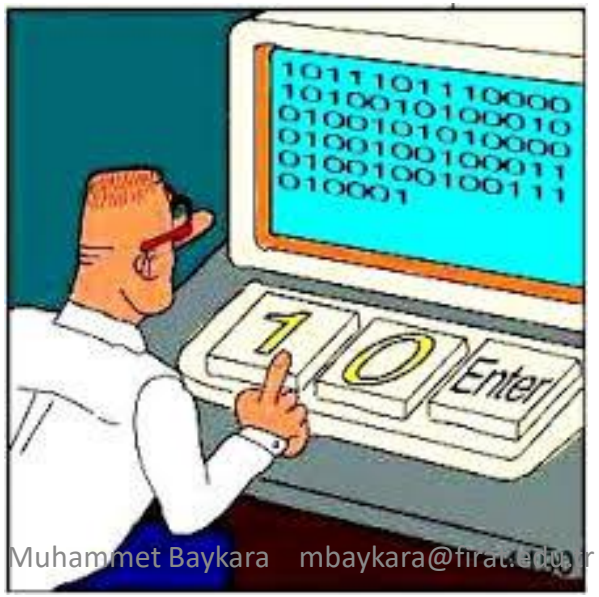
// ikincisi
// A class for keeping track of a series of doubles
// and methods for quick statistics about them.
class StatsKeeper {
  public:
    void Add(double d);
    double Average();

  private:
    list<double> past_items;
    int count; // how many so far

    double minimum;
    double maximum;
};
```

Programlama Dillerinin Tarihçesi

- İlk programlar fiziksel olarak yazılıyordu. Daha sonra fiziksel programlama yerini elektrik sinyaline bıraktı. Artık, kurulan elektronik devrelere düşük ya da yüksel voltajda akım gönderilerek bilgisayarın davranışı belirlenmeye başlandı. Yüksel voltaj 1, düşük voltaj 0 sayılarını ifade ediyordu.
- Böylelikle bugün de kullanılan makine dilinin ortaya çıkması için ilk adımlar atılmış oldu.



Programlama Dillerinin Tarihçesi

- Ancak bu şekilde programlar yazmak, sistemi oluşturan elektronik devrelerin her program için baştan kurulmasını gerektiriyordu. Bundan dolayı programlar bazı kavramlar çerçevesinde yazılmaya başlandı.
- Öncelikle bilgisayar donanımı her program için baştan kurulmamalı, bunun yerine basit bir donanımın üzerine yazılan komutlar kullanılmalıdır. Daha sonra, programlar tek bir komutlar zinciri yerine, küçük parçalar halinde yazılmalıdır.
- Bu parçaların programın içinde defalarca kullanılabilmesi yordam (subroutine) kavramını ortaya çıkarmıştır. Bu modelin kullanılması ise mantıksal karşılaştırmaları, döngülerin kullanılmasını ve yazılan kodlar tekrar kullanıldığı için kütüphane (library) mantığını ortaya çıkarmıştır.

Programlama Dillerinin Tarihçesi

- 1957 yılında IBM, düşük seviye (makine diline yakın) bir programlama dili olan FORTRAN dilini ortaya çıkardı.
- FORTRAN ile beraber basit mantıksal karşılaştırmalar, döngüler, (true-false) lojik ve (integer, double) sayısal değişkenler kullanılmaya başlandı.
- 1959 yılında, bu programlama dilinin özelliklerini alıp, giriş çıkış (Input – Output IO) gibi yeni işlevler sağlayan COBOL dili ortaya çıktı.
- Daha sonra 1968 yılında, COBOL ve FORTRAN dillerinin en iyi özelliklerini alarak Pascal ortaya çıktı. Ayrıca Pascal dili, hafızadaki adresler üzerinde işlem yapmaya olanak veren işaretçi (pointer) kavramını beraberinde getirdi.

Programlama Dillerinin Tarihçesi

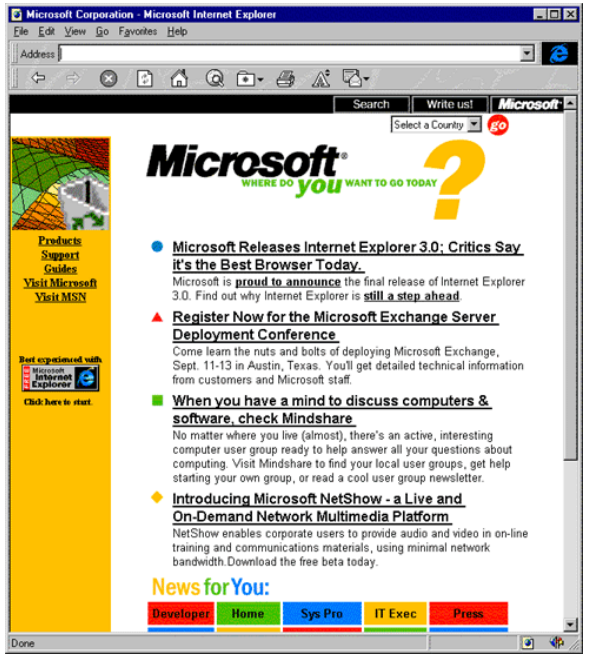
- 1972 yılında C, Pascal dilindeki birçok hatayı gidererek ortaya çıktı. C dili ilk defa Unix işletim sistemini yazmak için kullanılmaya başlanmıştır.
- C, orta seviye bir dil olması, kuvvetli giriş çıkış işlemleri sağlaması gibi birçok özelliği ile işletim sistemleri yazılmasında tercih edilmiştir.
- Bütün programlama dilleri birçok özelliğe sahip olmasına rağmen, modüler programlamanın birçok eksiğini gidermek amacıyla, yeni bir programlama modeli olan nesneye yönelik programlama - OOP (object oriented programming) ortaya çıkarıldı. C dilinin ve OOP modelinin tüm özellikleriyle C++dili oluşturdu

Programlama Dillerinin Tarihçesi

- C++ dilini, Sun Microsystems tarafından çıkartılan Java takip etti. Java dilinin kullanım alanları, nesneye yönelik bir programlama dili olması ve beraberinde getirdiği çöp toplama GC (garbage collection) gibi performans arttırıcı özellikleri ile büyük ölçüde genişledi.
- Microsoft, 2000 yılında .NET platformunu sunarak, otuzdan fazla programlama dilini aynı çatı altına topladı. VisualBasic.NET ve VisualC# .NET platformunu kullanan günümüzdeki en güçlü yüksek seviyeli programlama dilleri arasında yer almışlardır.

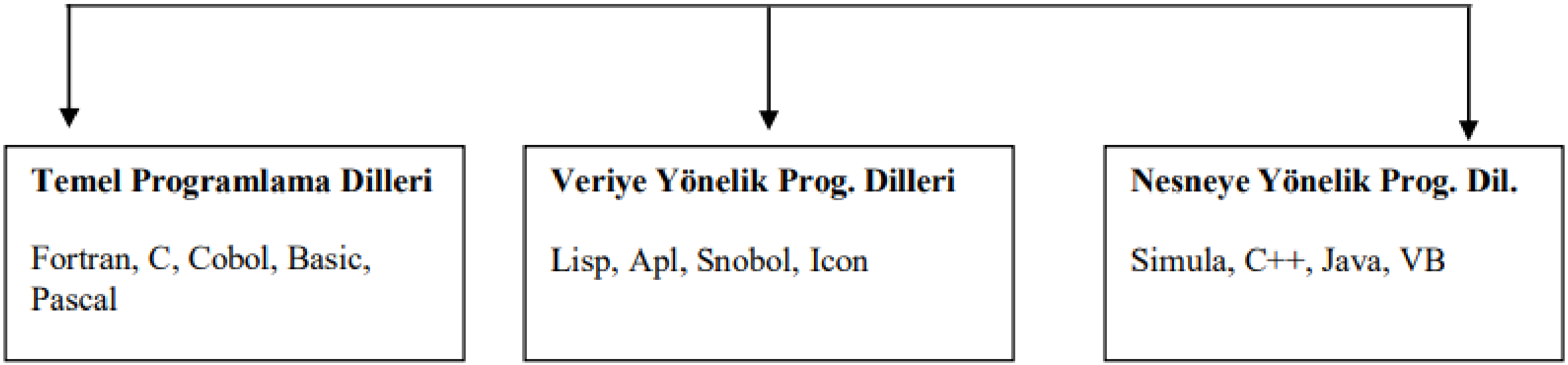
Programlama Dillerinin Tarihçesi

- C++ dilini, Sun Microsystems tarafından çıkartılan Java takip etti. Java dilinin kullanım alanları, nesneye yönelik bir programlama dili olması ve beraberinde getirdiği çöp toplama GC (garbage collection) gibi performans arttırıcı özellikleri ile büyük ölçüde genişledi.
- Microsoft, 2000 yılında .NET platformunu sunarak, otuzdan fazla programlama dilini aynı çatı altına topladı. VisualBasic.NET ve VisualC# .NET platformunu kullanan günümüzdeki en güçlü yüksek seviyeli programlama dilleri arasında yer almışlardır.



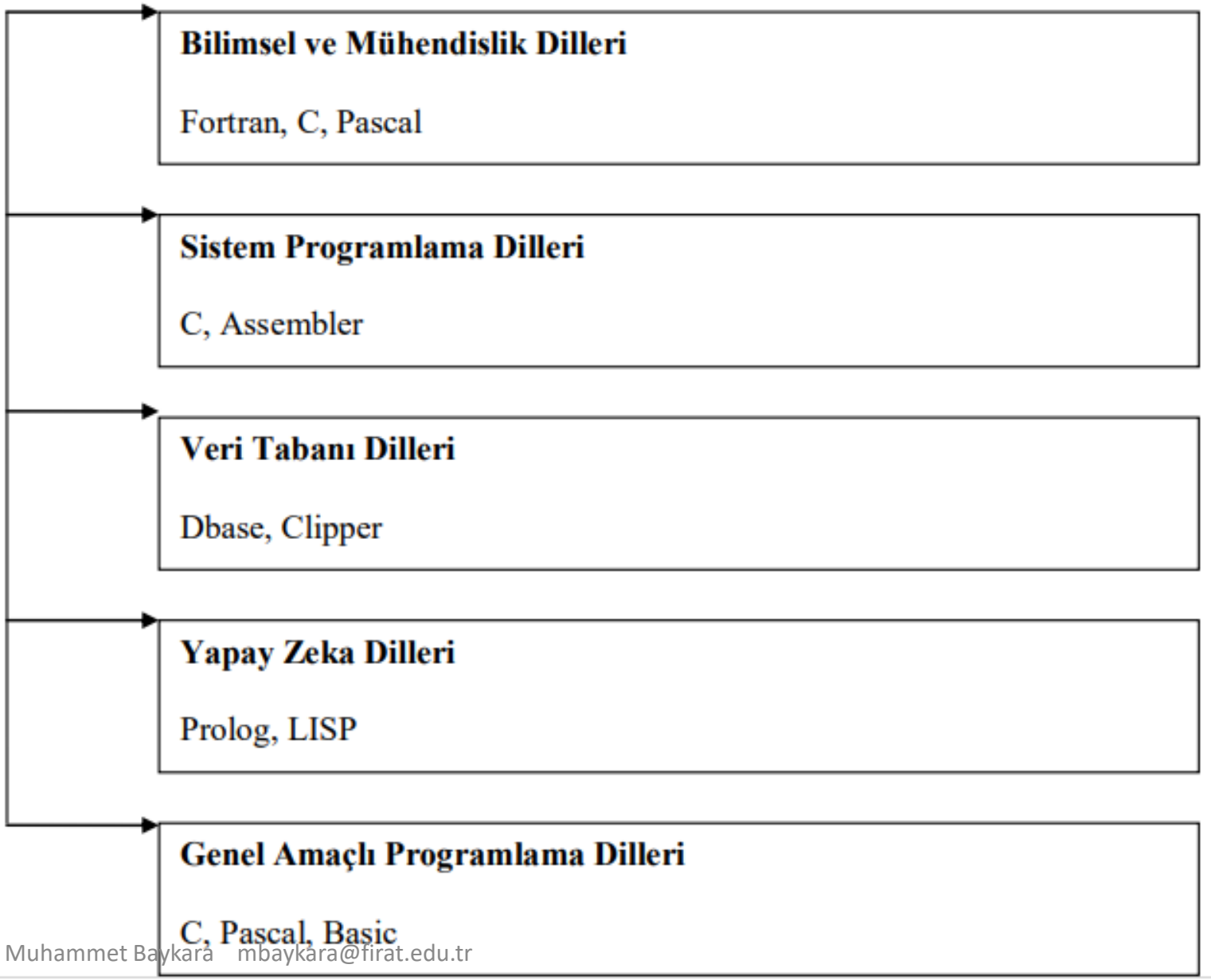
Programlama Dillerinin Sınıflandırılması

1- Genel Sınıflandırma



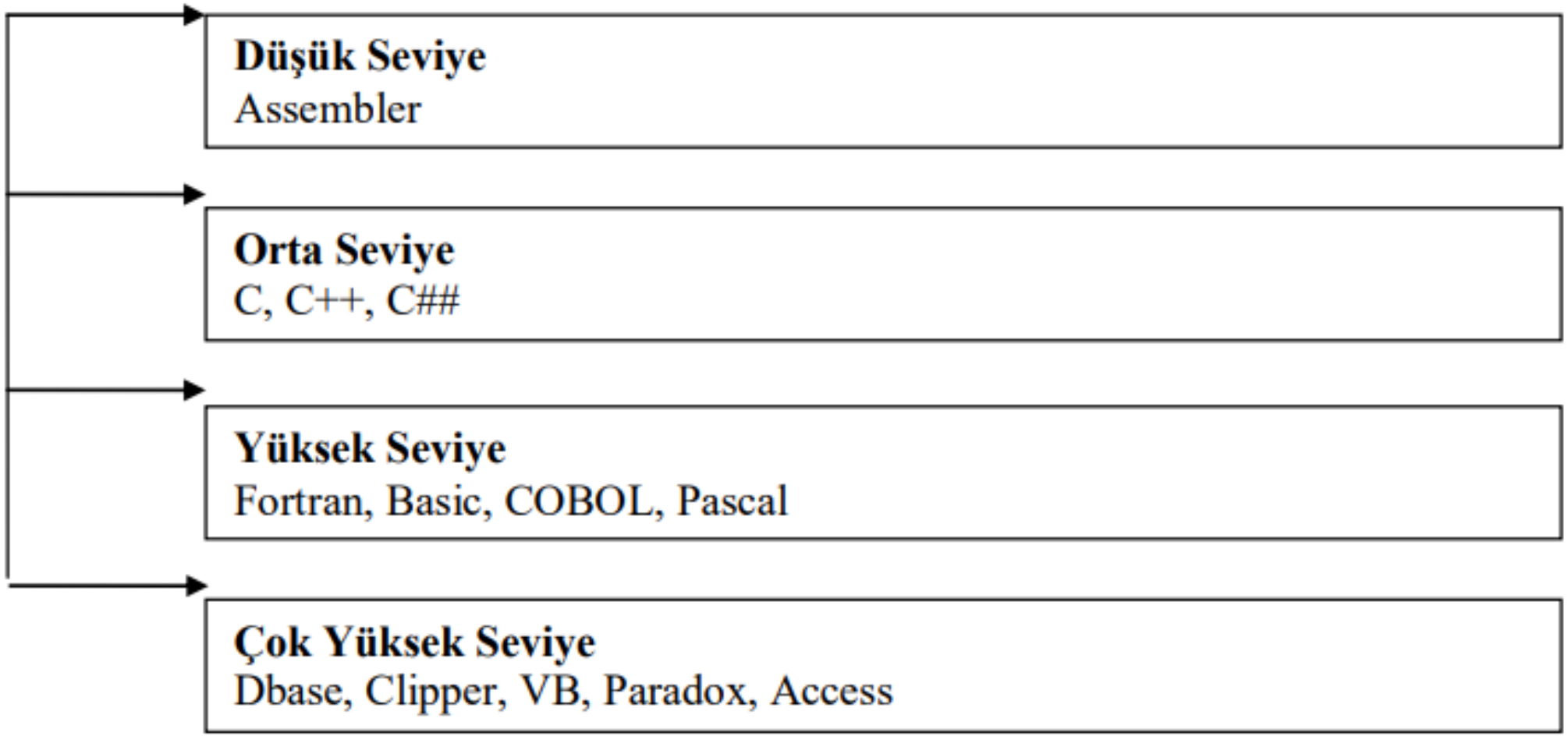
Programlama Dillerinin Sınıflandırılması

2- Uygulama Alanlarına Göre Sınıflandırma



Programlama Dillerinin Sınıflandırılması

3-Seviyelerine Göre Sınıflandırma



Nesneye Yönelik Programlama

- Nesneye yönelik programlama nesne kavramına dayanmaktadır. Burada nesne gerçek dünyada var olan veya programcı tarafından oluşturulmuş mantıksal bir varlıktır. Nesne, kendisini tanımlayan veriler ve bu veriler üzerinde yapılacak tüm işlemler ile bir bütün olarak düşünülür.
- **Veri Soyutlama (Data Abstraction) :** Kullanıcı tarafından yeni veri türlerini modelleyen sınıflar oluşturulmasıdır.
- **Kalıtım (Inheritance) :** Oluşturulan bu sınıfların genişleterek veya özelleştirilerek yeni sınıflar oluşturulmasıdır.
- **Çok Biçimlilik (Polymorphism) :** Aynı isimdeki işlemlerin değişik nesne grupları tarafından farklı algılanmasıdır.

Nesneye Yönelik Programlama Dilleri











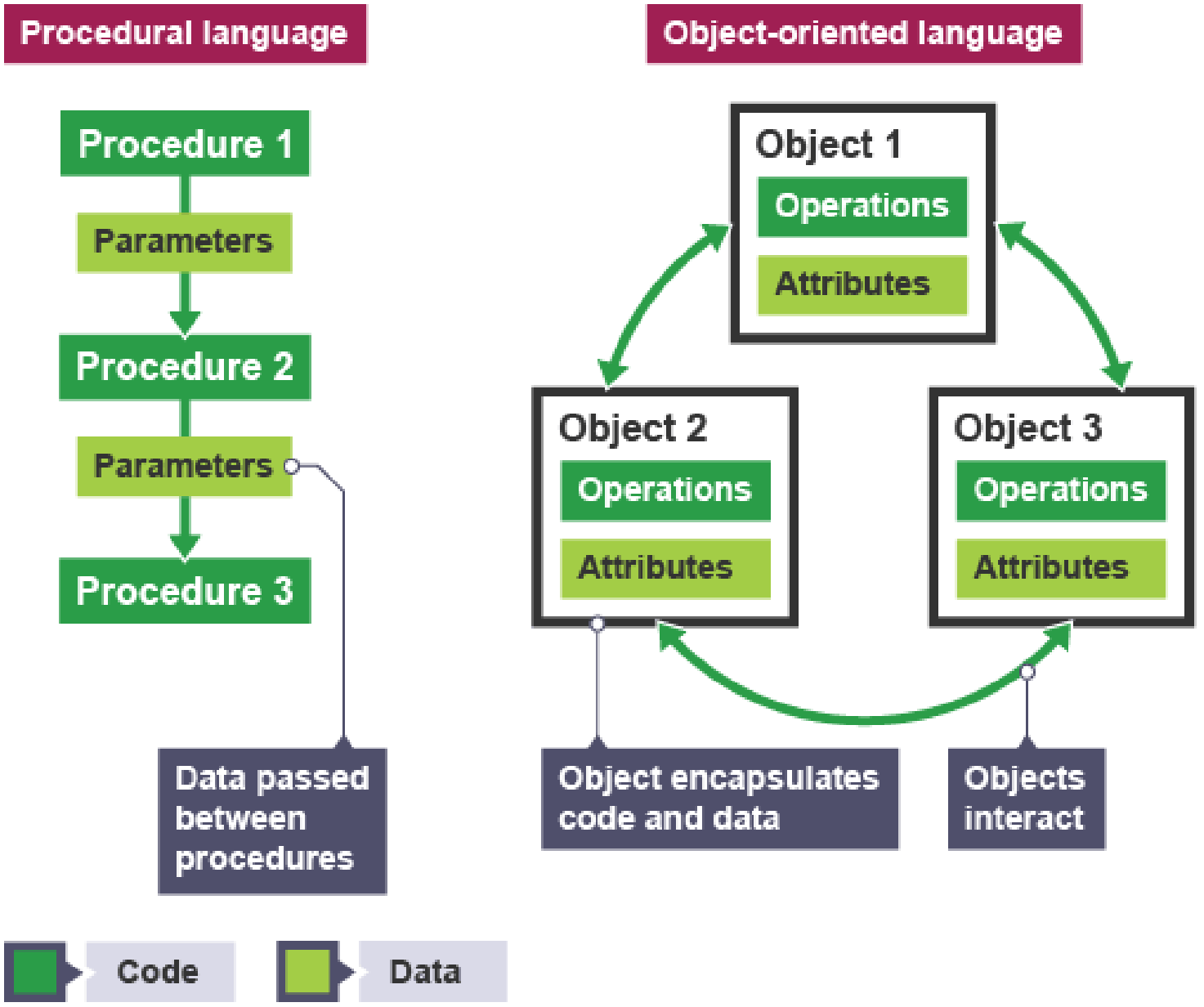








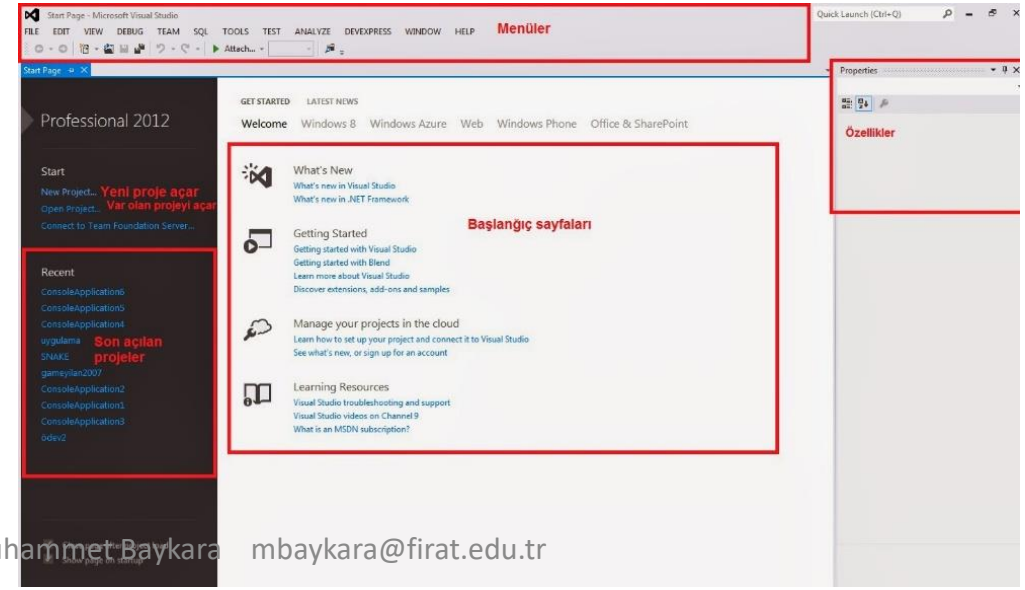
Muhammet Baykara mbaykara@firat.edu.tr

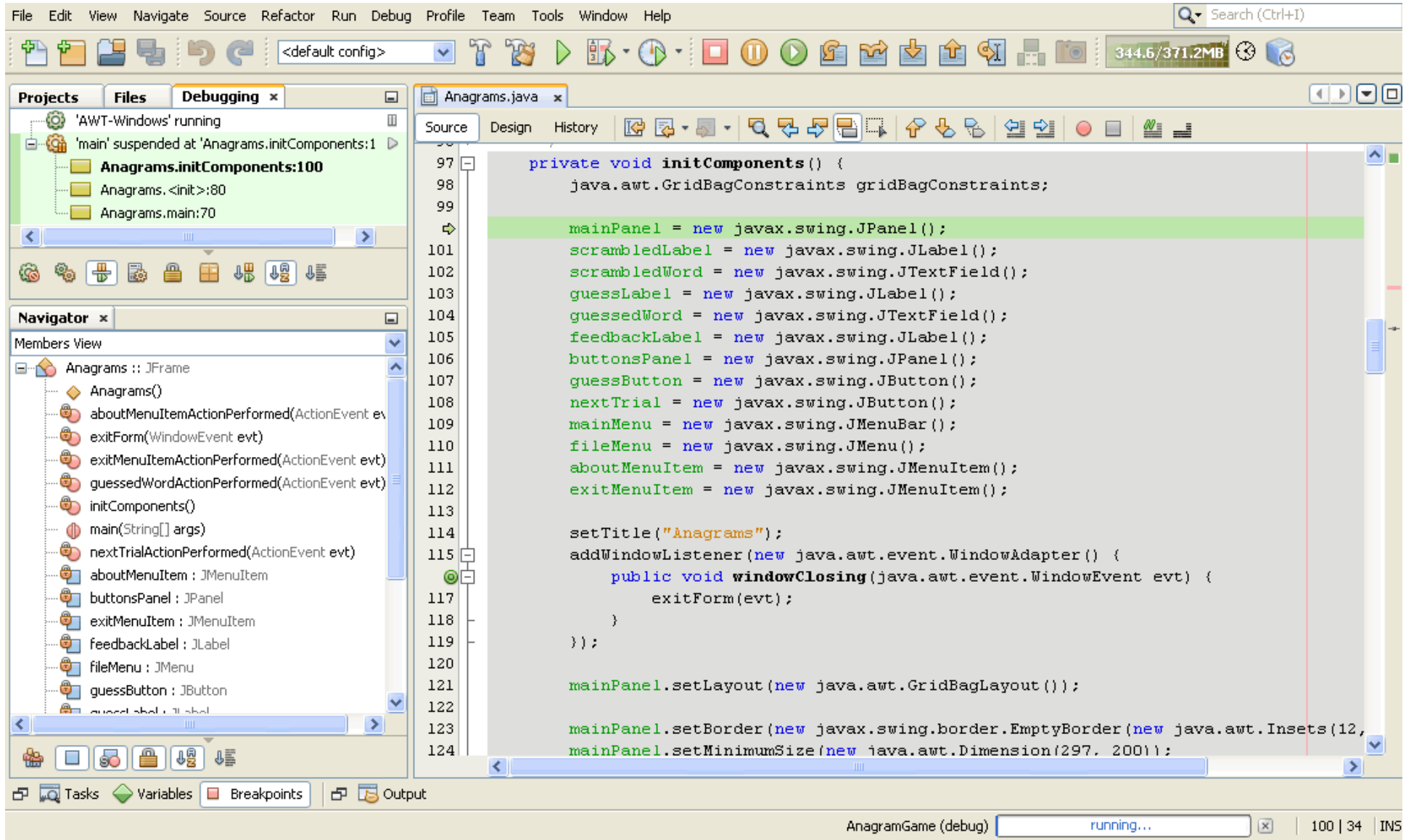


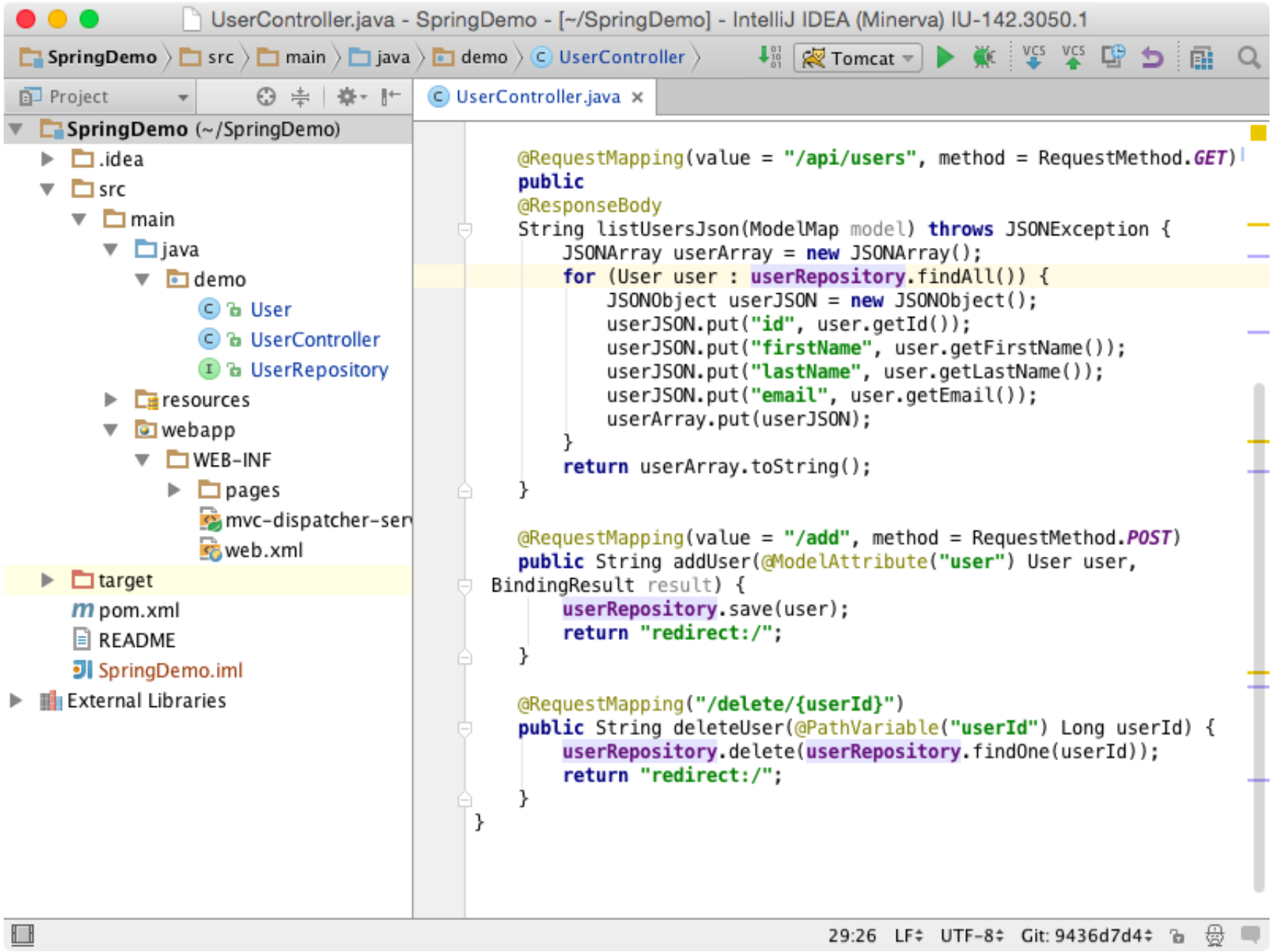
Muhammet Baykara mbaykara@firat.edu.tr

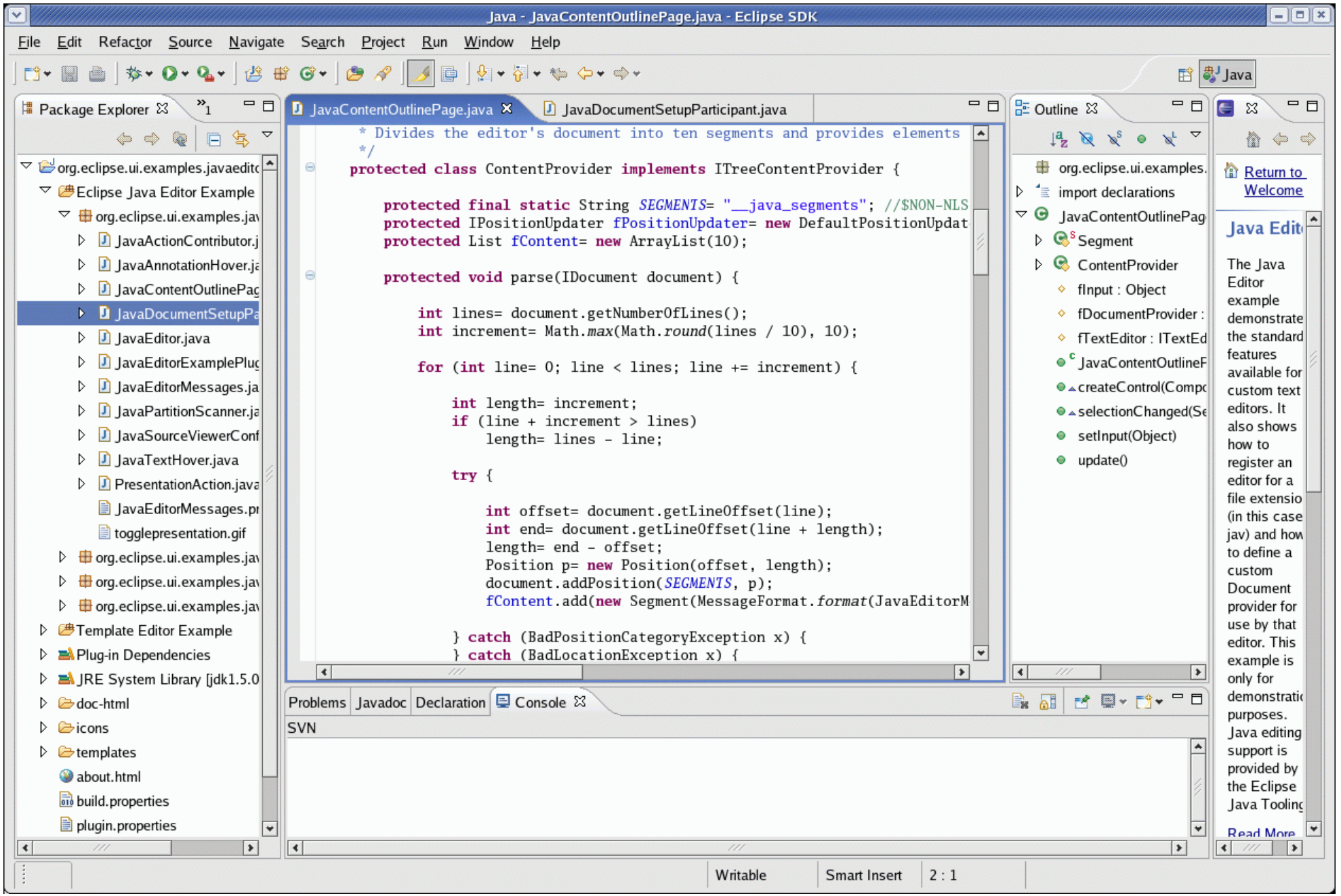
Programlama Ortamı

- Programlama ortamı, programlama dili ile birlikte birçok bileşenden oluşur.
- Bu bileşenler, sembolik olarak tasarlamış programın bilgisayar donanımı tarafından istenilen işlevleri yerine getirmesi için gereken tüm unsurlardır.
- Programlama ortamının temel unsurları Editör (Editor), Derleyici (Compiler), Kütüphane (Library), Bağlayıcı, Hata ayıklayıcı (Debugger) ve yorumlayıcı (Interpreter) dır.




File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
344.6/371.2MB
Projects Files Debugging
'AWT-Windows' running
'main' suspended at 'Anagrams.initComponents:1
Anagrams.initComponents:100
Anagrams.<init>:80
Anagrams.main:70
Navigator
Members View
Anagrams :: JFrame
Anagrams()
aboutMenuItemActionPerformed(ActionEvent ev
exitForm(WindowEvent evt)
exitMenuItemActionPerformed(ActionEvent evt)
guessedWordActionPerformed(ActionEvent evt)
initComponents()
main(String[] args)
nextTrialActionPerformed(ActionEvent evt)
aboutMenuItem : JMenuItem
buttonsPanel : JPanel
exitMenuItem : JMenuItem
feedbackLabel : JLabel
fileMenu : JMenu
guessButton : JButton
Anagrams.java
Source Design History
97 private void initComponents() {
98 java.awt.GridBagConstraints gridBagConstraints;
99
mainPanel = new javax.swing.JPanel();
101 scrambledLabel = new javax.swing.JLabel();
102 scrambledWord = new javax.swing.JTextField();
103 guessLabel = new javax.swing.JLabel();
104 guessedWord = new javax.swing.JTextField();
105 feedbackLabel = new javax.swing.JLabel();
106 buttonsPanel = new javax.swing.JPanel();
107 guessButton = new javax.swing.JButton();
108 nextTrial = new javax.swing.JButton();
109 mainMenu = new javax.swing.JMenuBar();
110 fileMenu = new javax.swing.JMenu();
111 aboutMenuItem = new javax.swing.JMenuItem();
112 exitMenuItem = new javax.swing.JMenuItem();
113
114 setTitle("Anagrams");
115 addWindowListener(new java.awt.event.WindowAdapter() {
public void windowClosing(java.awt.event.WindowEvent evt) {
117 exitForm(evt);
118 }
119 });
120
121 mainPanel.setLayout(new java.awt.GridBagLayout());
122
123 mainPanel.setBorder(new javax.swing.border.EmptyBorder(new java.awt.Insets(12,
124 mainPanel.setMinimumSize(new java.awt.Dimension(297, 200));
Tasks Variables Breakpoints Output
AnagramGame (debug) running... 100 | 34 INS

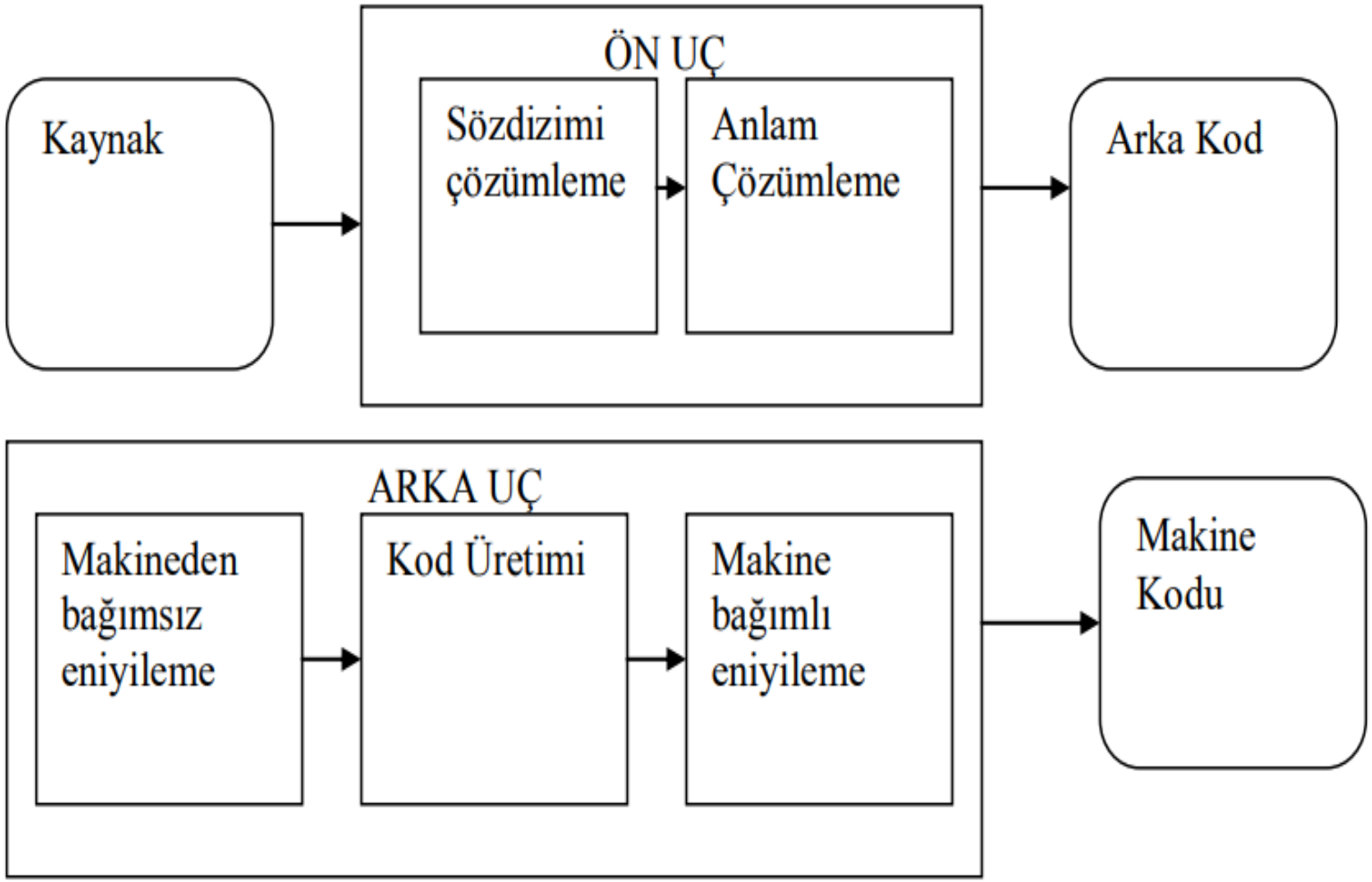

UserController.java - SpringDemo - [~/SpringDemo] - IntelliJ IDEA (Minerva) IU-142.3050.1
SpringDemo
src
main
java
demo
UserController
Tomcat
Project
UserController.java
SpringDemo (~/SpringDemo)
.idea
src
main
java
demo
User
UserController
UserRepository
resources
webapp
WEB-INF
pages
mvc-dispatcher-ser
web.xml
target
pom.xml
README
SpringDemo.iml
External Libraries
@RequestMapping(value = "/api/users", method = RequestMethod.GET)
public
@ResponseBody
String listUsersJson(ModelMap model) throws JSONException {
JSONArray userArray = new JSONArray();
for (User user : userRepository.findAll()) {
JSONObject userJSON = new JSONObject();
userJSON.put("id", user.getId());
userJSON.put("firstName", user.getFirstName());
userJSON.put("lastName", user.getLastName());
userJSON.put("email", user.getEmail());
userArray.put(userJSON);
}
return userArray.toString();
}
@RequestMapping(value = "/add", method = RequestMethod.POST)
public String addUser(@ModelAttribute("user") User user,
BindingResult result) {
userRepository.save(user);
return "redirect:/";
}
@RequestMapping("/delete/{userId}")
public String deleteUser(@PathVariable("userId") Long userId) {
userRepository.delete(userRepository.findOne(userId));
return "redirect:/";
}
}
29:26
LF
UTF-8
Git: 9436d7d4

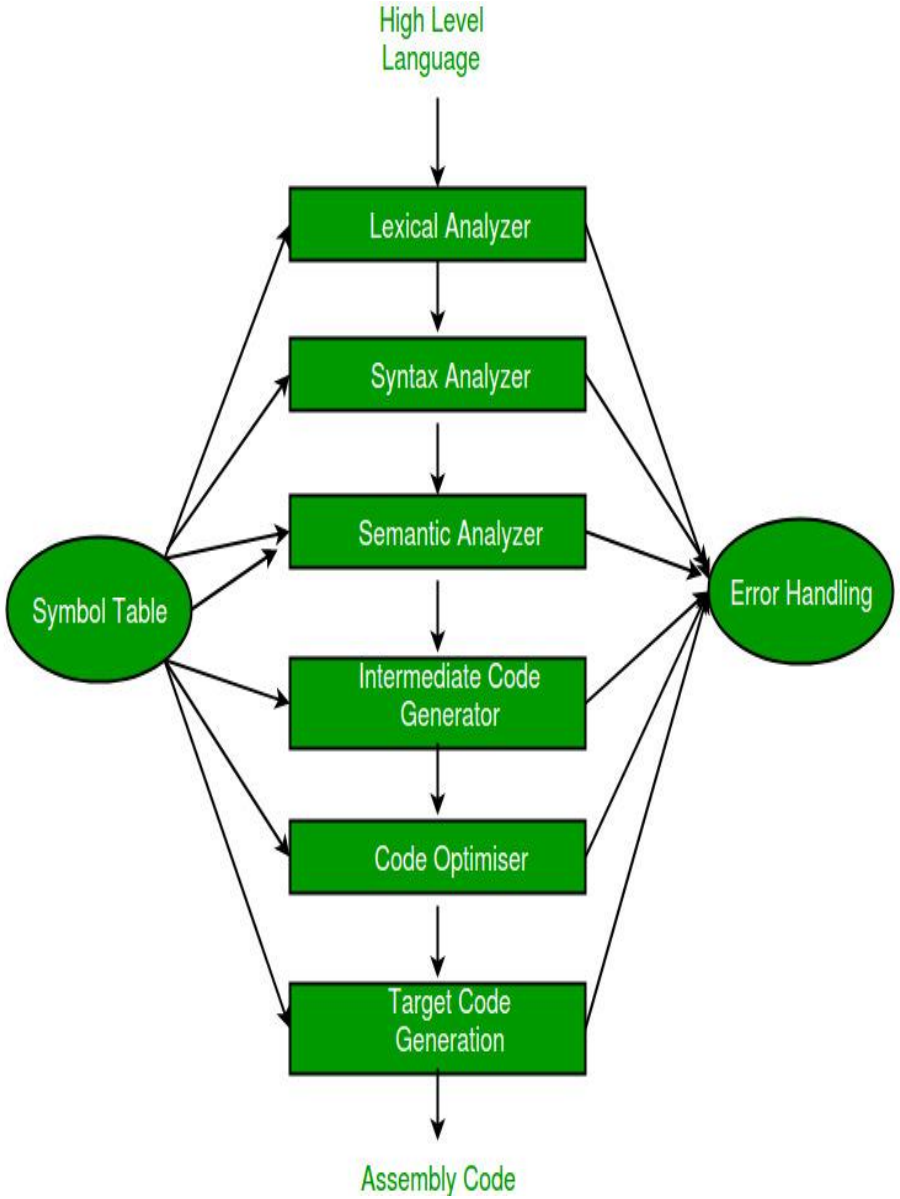

Java - JavaContentOutlinePage.java - Eclipse SDK
File Edit Refactor Source Navigate Search Project Run Window Help
Java
Package Explorer
org.eclipse.ui.examples.javaedit
Eclipse Java Editor Example
org.eclipse.ui.examples.jav
JavaActionContributor.j
JavaAnnotationHover.ja
JavaContentOutlinePag
JavaDocumentSetupPa
JavaEditor.java
JavaEditorExamplePlug
JavaEditorMessages.ja
JavaPartitionScanner.ja
JavaSourceViewerConf
JavaTextHover.java
PresentationAction.java
JavaEditorMessages.pr
togglepresentation.gif
org.eclipse.ui.examples.jav
org.eclipse.ui.examples.jav
org.eclipse.ui.examples.jav
Template Editor Example
Plug-in Dependencies
JRE System Library [jdk1.5.0
doc-html
icons
templates
about.html
build.properties
plugin.properties
JavaContentOutlinePage.java
JavaDocumentSetupParticipant.java
* Divides the editor's document into ten segments and provides elements
*/
protected class ContentProvider implements ITreeContentProvider {
protected final static String SEGMENTS= "__java_segments"; //$NON-NLS
protected IPositionUpdater fPositionUpdater= new DefaultPositionUpdat
protected List fContent= new ArrayList(10);
protected void parse(IDocument document) {
int lines= document.getNumberOfLines();
int increment= Math.max(Math.round(lines / 10), 10);
for (int line= 0; line < lines; line += increment) {
int length= increment;
if (line + increment > lines)
length= lines - line;
try {
int offset= document.getLineOffset(line);
int end= document.getLineOffset(line + length);
length= end - offset;
Position p= new Position(offset, length);
document.addPosition(SEGMENTS, p);
fContent.add(new Segment(MessageFormat.format(JavaEditorM
} catch (BadPositionCategoryException x) {
} catch (BadLocationException x) {
Outline
org.eclipse.ui.examples.
import declarations
JavaContentOutlinePag
Segment
ContentProvider
fInput : Object
fDocumentProvider :
fTextEditor : ITextEd
JavaContentOutlineF
createControl(Compc
selectionChanged(Se
setInput(Object)
update()
Return to Welcome
Java Edit
The Java Editor example demonstrate the standard features available for custom text editors. It also shows how to register an editor for a file extensio (in this case jav) and how to define a custom Document provider for use by that editor. This example is only for demonstrati purposes. Java editing support is provided by the Eclipse Java Tooling
Read More
Problems Javadoc Declaration Console
SVN
Writable Smart Insert 2 : 1

Programlama Ortamı

- **Editör (Editor):** Kaynak kodu oluşturmak ve gerektiğinde değişiklik yapmak için gerekli olan araçtır. Editörde yazılanlar seçilen dilin komutlarından oluşan metinlerdir.
- **Derleyici (Compiler):** Editörde yazılan kaynak kodu makine koduna çeviren bir programdır.
- **Kütüphane (Library):** Nesne dosyalarından oluşur.
- **Bağlayıcı:** Programın içerdiği tüm nesne dosyalarını birleştirerek tek dosya haline getirerek yürütülebilir bir dosya haline getirir.
- **Hata ayıklayıcı (Debugger):** Programcının hataları ayıklayabilmesi için programın adım adım yürütülmesini sağlar.
- **Yorumlayıcı (Interpreter):** Programın kaynak kodunu doğrudan satır satır yürüten bir programdır.

Bir Derleyicinin Genel Yapısı





Programlama Dillerinin Elemanları

- **Sözdizimi/Yazım (Syntax):** Temel olarak bir dilde (language) tanımlı olan öğelerin (kelime, işlem, sembol yada değerlerin) anlamlı bir dizilim oluşturmasıyla ilgilenen bilimdir.

- **Anlambilim (Semantics):** Bir programlama dilindeki bir ifadenin ne anlama geldiğidir.

- **Veri (Data):** Bir programda farklı veri tipleriyle işlem yapmamız gerekebilir. Örneğin, tamsayılar, kesirli sayılar, karakterler (harfler ve klavyedeki diğer simgeler), metinler (string), mantıksal (boolean) değerler (doğru=true, yanlış=false) ilk aklımıza gelen farklı veri tipleridir

Programlama Dillerinin Elemanları

- **Atama Deyimi (Assignment Statement):** Şeklindeki satırlar birer atama deyimini (assignment statement) göstermektedir. Bu atama deyimleri ile x değişkeninin bellekteki adresinde 5 değeri ve y değişkeninin bellekteki adresinde ise 6 değeri görünecektir.

```
1 | x=5;
2 |
3 | y=6;
```

- **Tür Kontrolü:** Programlama dillerinde yapılan her işlem öncesi hataları önlemek için verilerin tip kontrolü yapılır. Bu tür kontrollere ”Checking” adı verilir. Checking işlemleri compile-time (derleme esnasında) veya run-time (çalışma esnasında) olarak yapılır. İşte bu farklılık Statik Tipli ve Dinamik Tipli dillerin temel ayrım noktalarından biridir.

Programlama Dillerinin Elemanları

- **Kontrol Deyimleri:** Program içerisinde bir işin olumlu veya olumsuz sonucuna göre yapılması gereken işlemler olabilir. Bu aşamada devreye kontrol deyimleri girer. Örnek kontrol deyimleri; If Else Deyimi, Switch-Case Deyimi.

- **Alt Programlar:** Bir programda aynı tür hesaplama işlemi programın farklı yer(ler)inde birden fazla kullanılabilir. Aynı işlem adımlarını bir çok kez tekrarlamak, programdaki deyim sayısını arttıracağından hem programın yavaş işlemesini, özellikle programı(n çalışması açısından) izlemeyi güçleştirir. Bu tür tekrarlanan program parçaları, ALT PROGRAM adı altında (ana programdan) ayrı bir program olarak yazılabilir.

```
public void denemeFonksiyonu(){
  if( s.toString().toLowerCase().indexOf("plane")>0){
    System.out.println("yer tıklandı");
  }
  else if(tg!=null){
    for(int i = 0 ;i<10;i++){
      if(tg.toString().equals(objectList[i])){
        tf.setText("");tf.setText(tf.getText()+"\n"+"Tank "+i+" tıklandı");
        tf.setText(tf.getText()+"\n"+"Tankın bölgesi :"+ tankRegion[i]);
        clickedTank = i;
        tf.setText(tf.getText()+"\n"+"Tank Owner : " + tankOwner[i] );

      }
    }
        if(tankOwner[clickedTank]!=activeplayer){
          tf.setText("");tf.setText(tf.getText()+"\n"+"Düşman tankı tıklanamaz");
          clickedTank=-1;
          return;
        }
        if(tankAction[clickedTank]<1){
          tf.setText("\n"+"Tank bu sene başka yere gidemez!"+tf.getText());
          clickedTank=-1;
          return;
        }
  }
 }
public static void main(String args[]{
  denemeFonksiyonu();
}
```

F
G
E
B
A
C
D
H

Muhammet Baykara mbaykara@firat.edu.tr

Programlama Dillerinin Elemanları

- **Modüller**: programlama (modular programming) tekniğinde, birbirleriyle ilgili olduğu düşünülen ya da aynı çağrışım kümesi içinde olduğu düşünülen fonksiyonlar bir araya toplanarak genellikle farklı bir dosya içinde tutulurlar.



Kaynaklar

http://content.lms.sabis.sakarya.edu.tr/Uploads/50101/39475/bmg_programlamadilleri.pdf
http://www.farukkirmizi.com/programlama-ve-programlama-ortami.html
https://algoritmaveprogramlama.wordpress.com/2013/09/21/programlama-terimleri-ve-programlama-ortami/
https://www.baskent.edu.tr/~tkaracay/etudio/ders/prg/java/ch05/dataTypes.htm
https://mhmtyylc.com/c-sharp-kontrol-deyimleri
https://acikders.ankara.edu.tr/pluginfile.php/20859/mod_resource/content/1/fzm205_8.pdf

Bilgisayar Ağları (Network)

Giriş

(SD0140)

Murat KARA

2017

1. Hafta

TANIŞMA, GİRİŞ, NETWORK(AĞ) NEDİR?

Merhaba

Kimim?	• Murat KARA
Neden bu ders?	• Ağ yapısını anlamak için,
Ne işe yarar?	• İletişimi tanımak,
Farkındalık gerekli mi?	• Bilgi ve Eğitim şart..
‘Uzman mı olucaz’?	• Öğreneceğiz.. Belki..
Uygulama yapacak mıyız?	• Sadece Lab. Ortamında.
Teknoloji ve Terminoloji?	• Pek çok..

*http://***193.140.54.45/network**
*http://***tik.la/zG8**
*http://***www.muratkara.com/network**

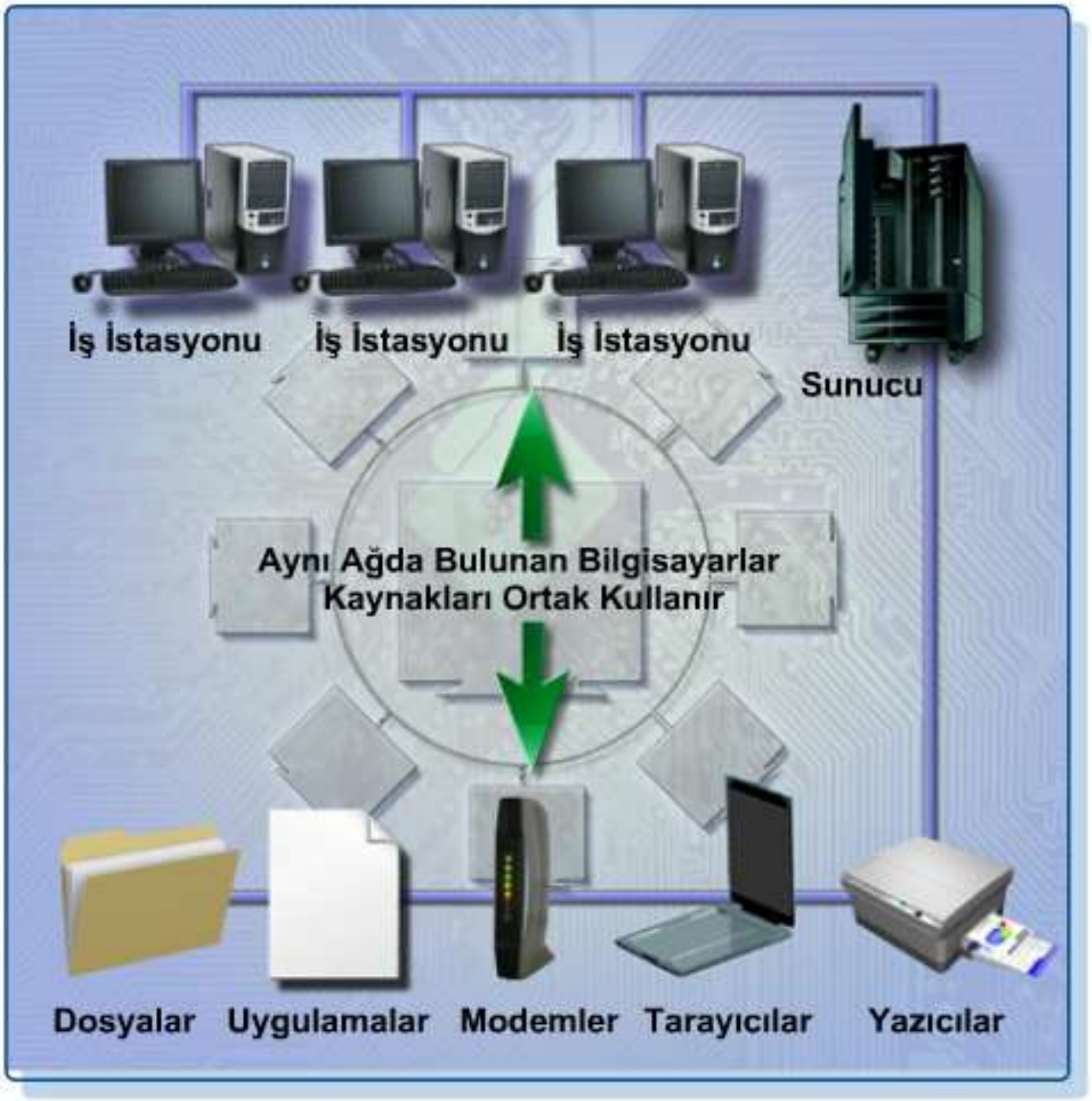
İçerik (Hafta/Konu)

1. Tanışma, bilgilendirme ve derse giriş, ağ nedir? Kullanım yerleri, sinyal, iletim.
2. Veri iletimi, temel özellikler ve temel terim ve kavramlar.
3. Referans modelleri (OSI, TCP/IP) ve katmanları..
4. Ağ Bileşenleri ve Ağların Sınıflandırılması (LAN, MAN, WAN..), Topolojiler..
5. Ağ Donanımları (Cihazları), Kablolama Sınıflamaları ve yapıları..
6. Protokoller, Protokol kümeleri ve Internet İletişimi (http, IP, dhcp, smtp, arp..)
7. Yerel Alan Ağ Teknolojileri (LAN, 802.x..)
8. Vize
9. Geniş Alan Ağ Teknolojileri (WAN, Anahtarlama, Yönlendirme..)
10. IP ve sınıflandırması IPv4-IPv6, alt ağlar(subnet),
11. Anahtar(Switch) Cihazı ve Anahtarlama yöntemleri..
12. Yönlendirici(Router) ve Yönlendirme çeşitleri..
13. Kablosuz Ağlar..
14. Ağ Güvenliği Saldırı ve Savunma Yöntemleri
15. Genel Tekrar ve Finale Hazırlık..

Network (Ağ)

- Ne?
- Neden?
- Nerede?
- Kullanım Yerleri?
- Neler?
- Nasıl?




İş İstasyonu
İş İstasyonu
İş İstasyonu
Sunucu
Aynı Ağda Bulunan Bilgisayarlar
Kaynakları Ortak Kullanır
Dosyalar
Uygulamalar
Modemler
Tarayıcılar
Yazıcılar

Ağ Kullanım Yerleri

- İletişim,
- Veri paylaşımı,
- Haberleşme,
- İnternet,
- Ticaret,
- Bankacılık,
- Eğitim
- …

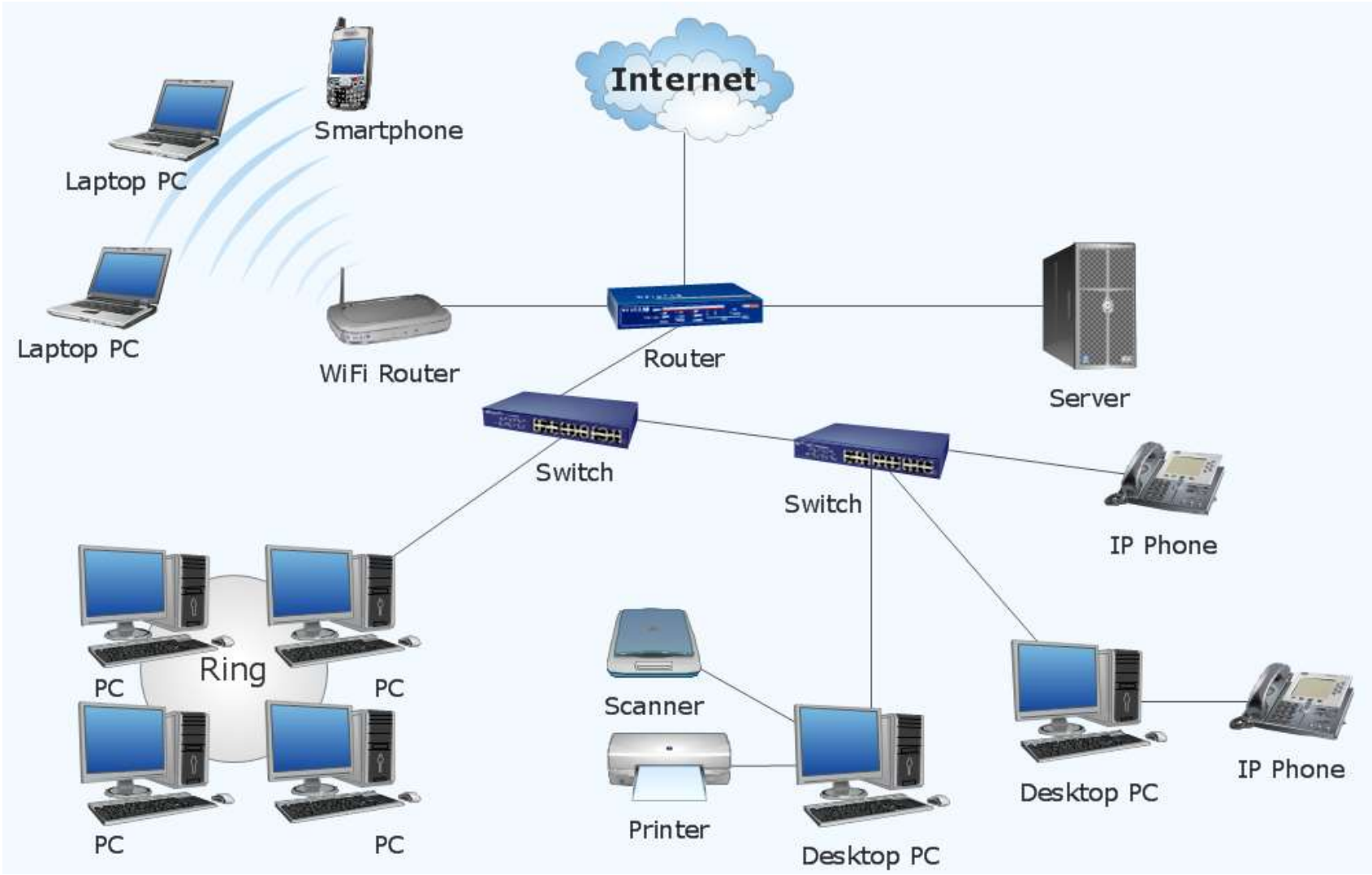


Bilgisayar Ağı (Network) Nedir?

- İki veya daha fazla bilgisayarın kablolu ya da kablosuz iletişim araçları üzerinden yazılım ve donanım bileşenleri ile birlikte bağlanması ile meydana getirilen sistem bütünüdür.

Bilgisayar Ağı (Network) Nedir?

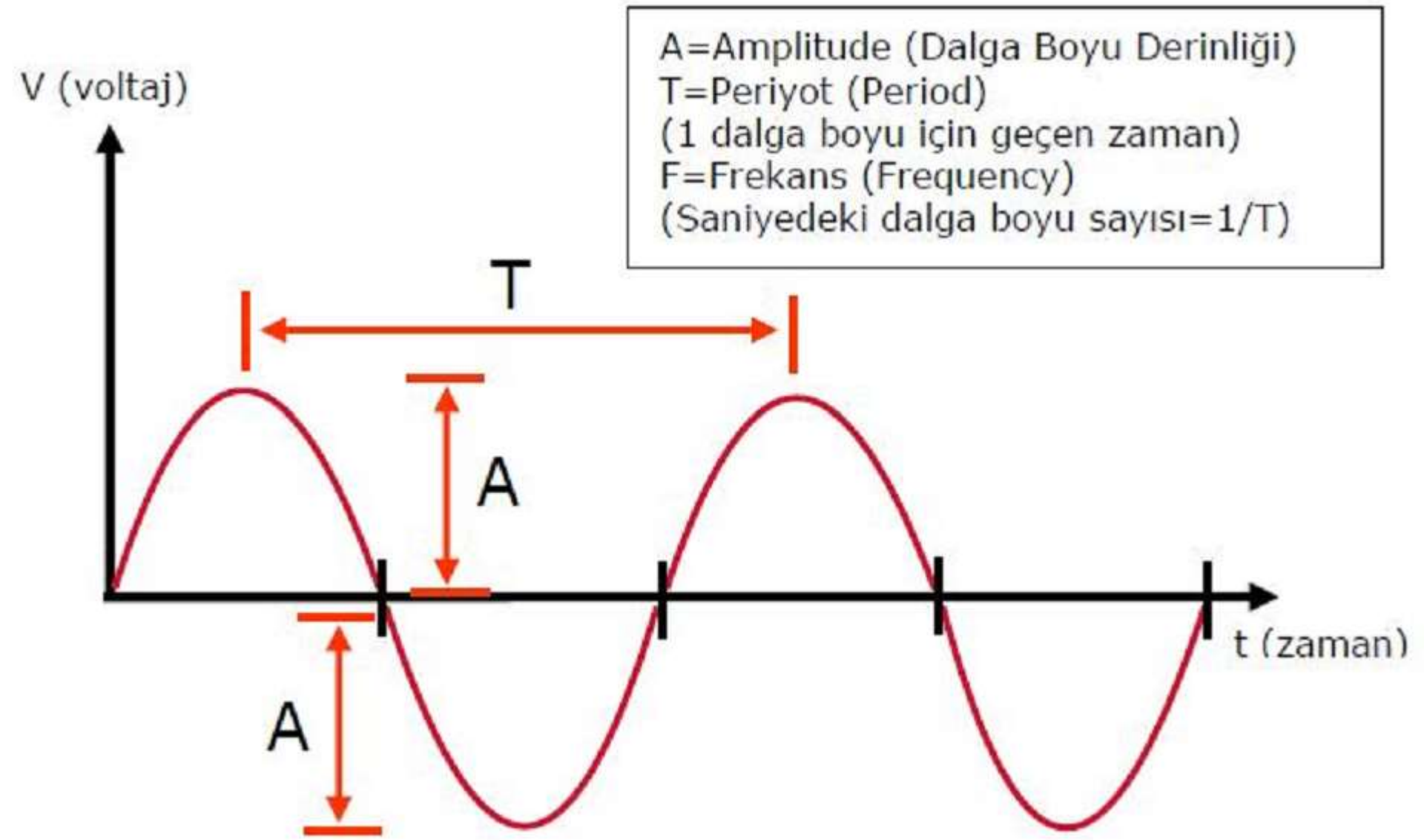
- Kaynakların ve bilginin (veri, ses, görüntü veya video) paylaşılması ve kişiler arasında iletişimin sağlanmasıdır.
- Çeşitli yöntem ve teknolojiler ile bağlanarak bilgisayar ağını oluştururlar.
- Kaynakların etkin paylaşımını sağlayıp bilgi akışını hızlandırarak verimli bir iletişim ortamı sunar.


Internet
Smartphone
Laptop PC
Laptop PC
WiFi Router
Router
Server
Switch
Switch
IP Phone
Ring
PC
PC
PC
PC
Scanner
Printer
Desktop PC
Desktop PC
IP Phone

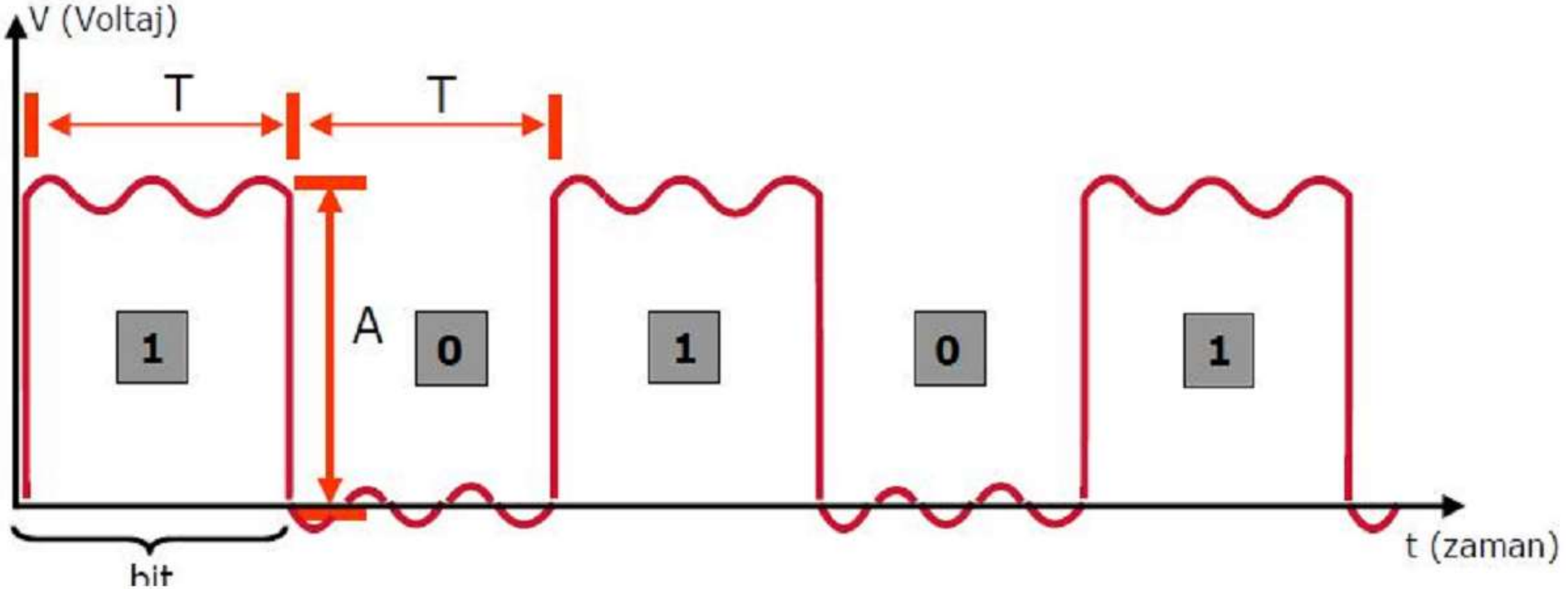
İletim, Sinyal..

- Sayısal iletişim ikili tabanda kodlanmış bilgi veya verinin sistemler arasında aktarılmasını kapsar.
- İletim
  - Paralel
  - Seri
    - Asenkron
    - Senkron
- Sinyal (İşaret), fiziksel değişkenlerin durumu hakkında bilgi taşıyan ve matematiksel olarak fonksiyon (İşlev) biçiminde gösterilen kavrama denir.

Analog Sinyal



Sayısal(Digital) Sinyal



2. Hafta

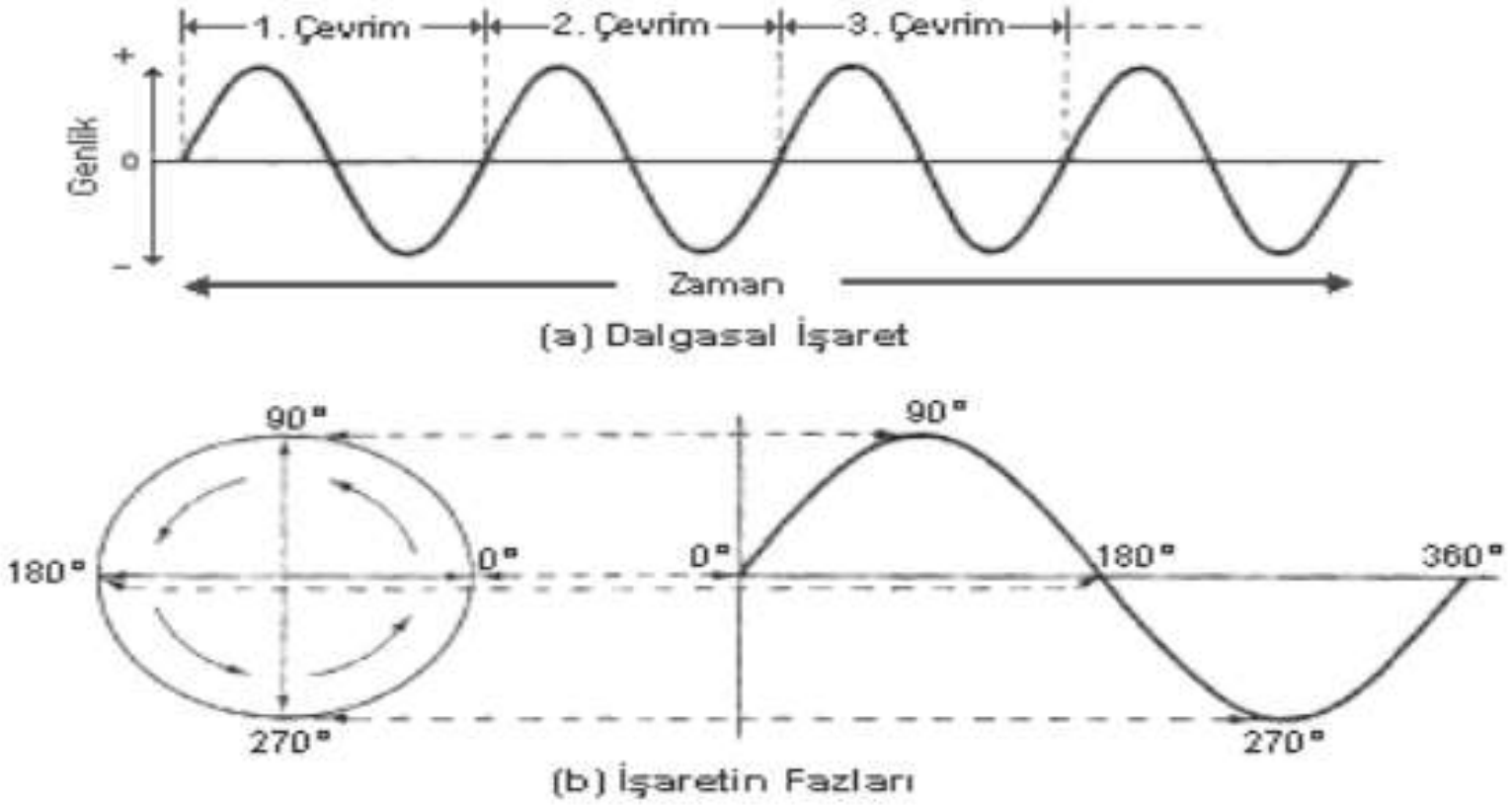
VERİ İLETİMİ, TEMEL TERİM VE KAVRAMLAR

Bilgisayarda verilerin taşınmasını sağlamak için salınım yapan işaretlerin üç parametresi değişebilir .

Genlik : kablo üzerine düşen elektrik yükü miktarı

Frekans: birim saniyedeki çevrim sayısıdır

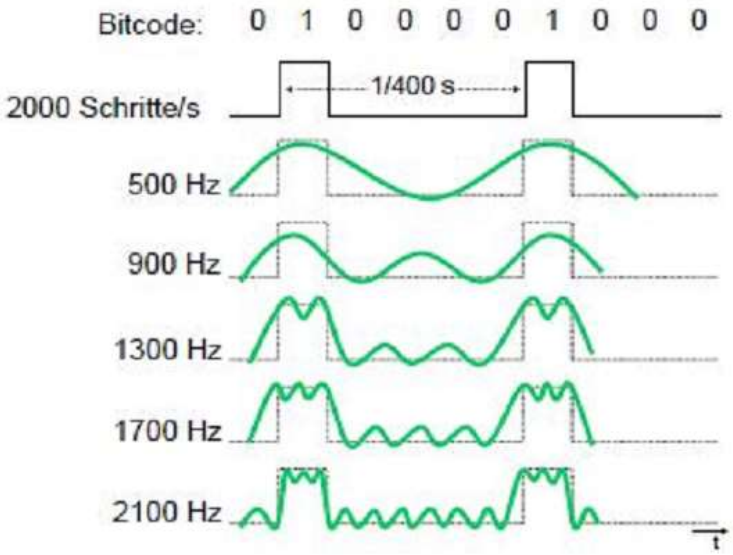
Faz: İşaretin çevrimine ne kadar ilerden başlayacağı

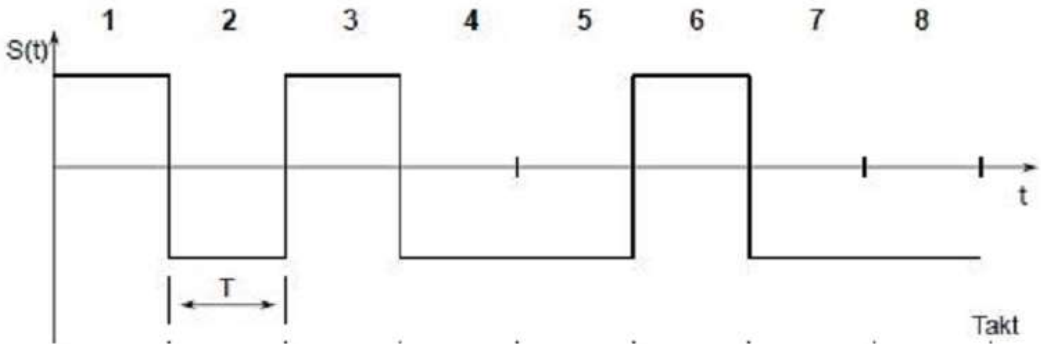


Şekil 1-3 Salınım Yapan İşaret

Frekans ve salınım

- Frekans arttıkça salınım artmakta ve daha fazla bilgi taşıyabiliriz.





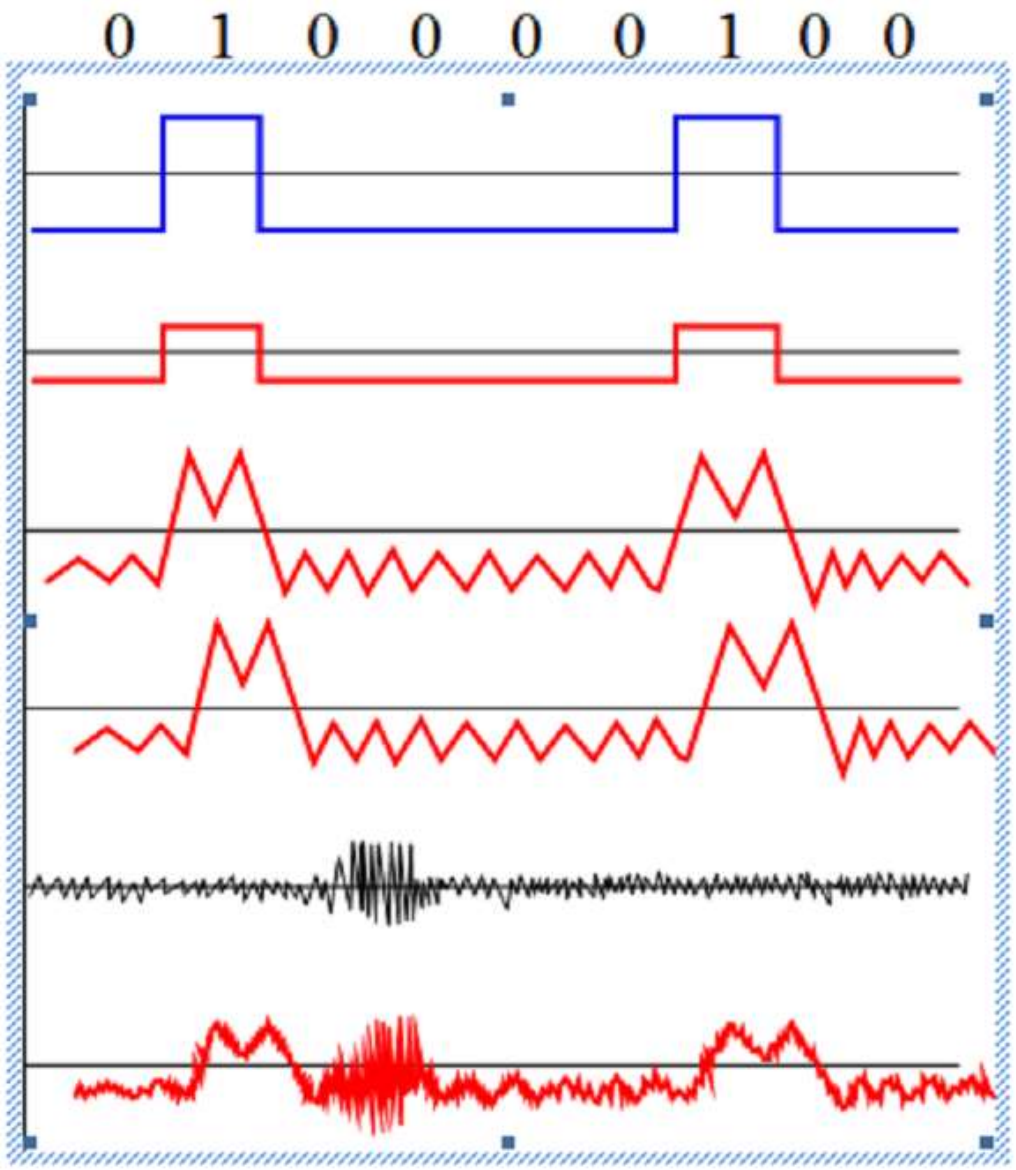
Terimler

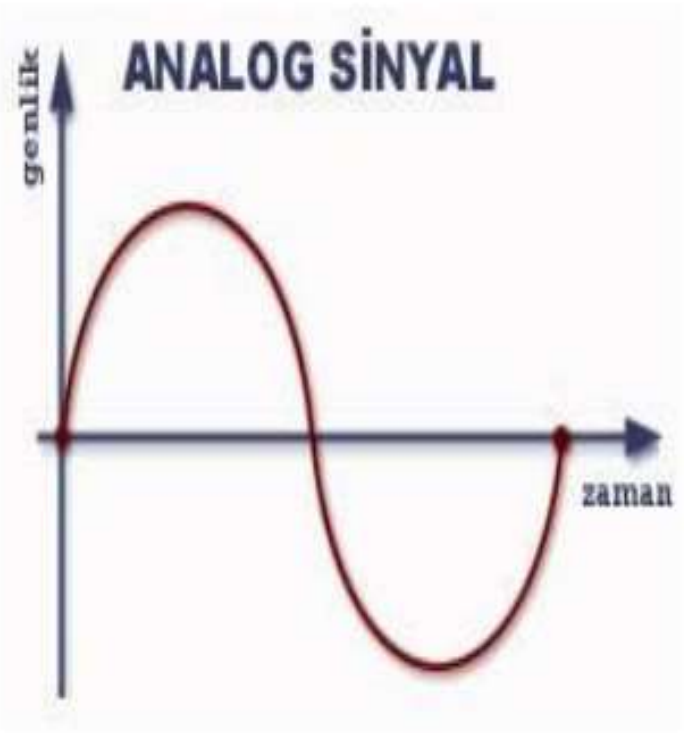
- Yayın (Propagation): Bir bilgisayarın gönderdiği bir bitlik datanın ağ ortamına bırakılıp hedef bilgisayara ulaşmasıdır.
- Gidiş-geliş zamanı (Round Trip Time-RTT): Bir bilgisayarın gönderdiği bir bitlik datanın ağ ortamına bırakılıp hedef bilgisayara ulaşıp kaynak bilgisayara geri gelme süresine denir.
- Yayın gecikmesi (Propagation Delay): Gönderilen datanın hedefe ulaşana kadar geçen süreye denir.
- Zayıflama (Attenuation): Dijital data bırakıldığı ağ ortamı ne olursa olsun kullanılan ortamın özelliğine göre belli bir mesafeden sonra zayıflayacaktır.
- Yansıma (Reflection): Elektrik sinyalleri devamlı olarak sinyalleşme sürdüğü müddetçe bir problem olmamakta ama sinyalleşme esnasında olan kopukluklarda geri bir sinyal alınmaktadır.

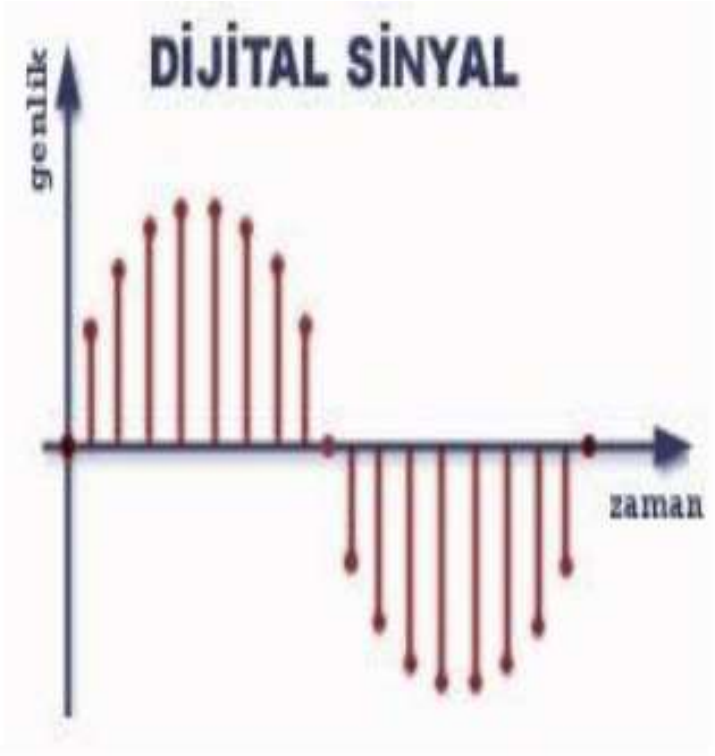
Terimler

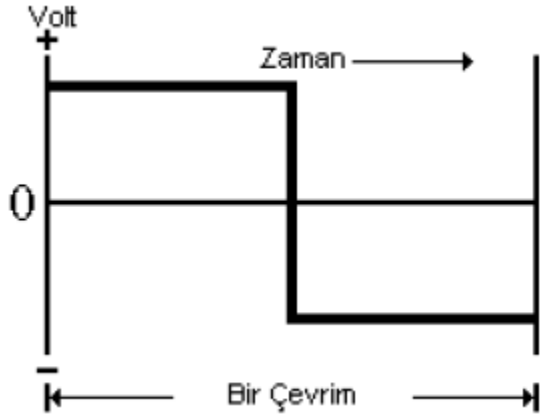
- Gürültü (Noise): Elektrik, optik ve elektromanyetik sinyallerin yakınında bulunan ve gönderilen datayı bozma riski olan diğer sinyallerin tümüne denir.
- Crosstalk: Bir kablonun başka bir kablonun elektrik sinyalinden veya elektriksel gürültüsünden etkilenmesine denir.
- Yayılma (Dispersion): Sinyalin zamana bağlı olarak yayılmasına denir.
- Jitter: Kaynak ve hedef bilgisayarların zamanlandırmasındaki farklılığa denir. (Paketlerin gecikme sürelerindeki varyasyondur-farklılıktır.
- Gecikme (Latency): Gönderilen datanın kaynak bilgisayardan hedef bilgisayara ulaşırken geçirdiği süreye denir.
- Çarpışma (Collision): İki bilgisayardan gönderilen bitlerin ağ ortamında çarpışmalarına denir.

Sinyal Görünümü

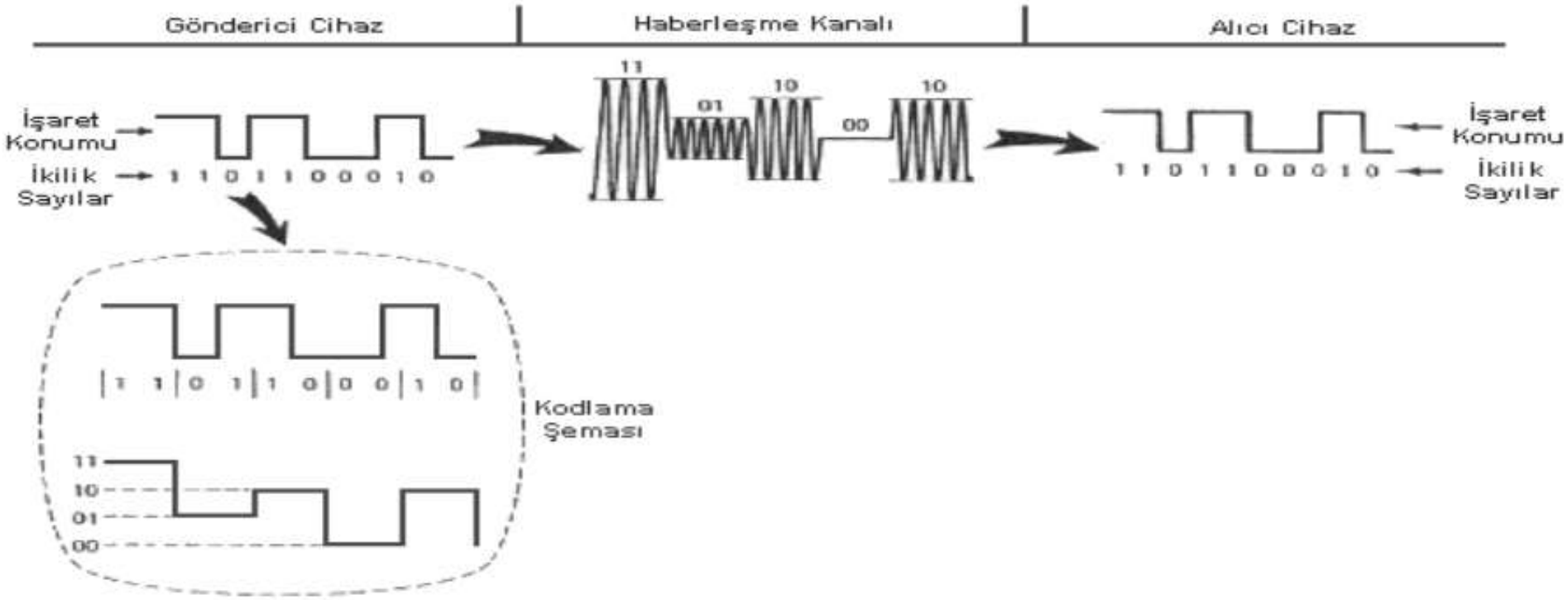






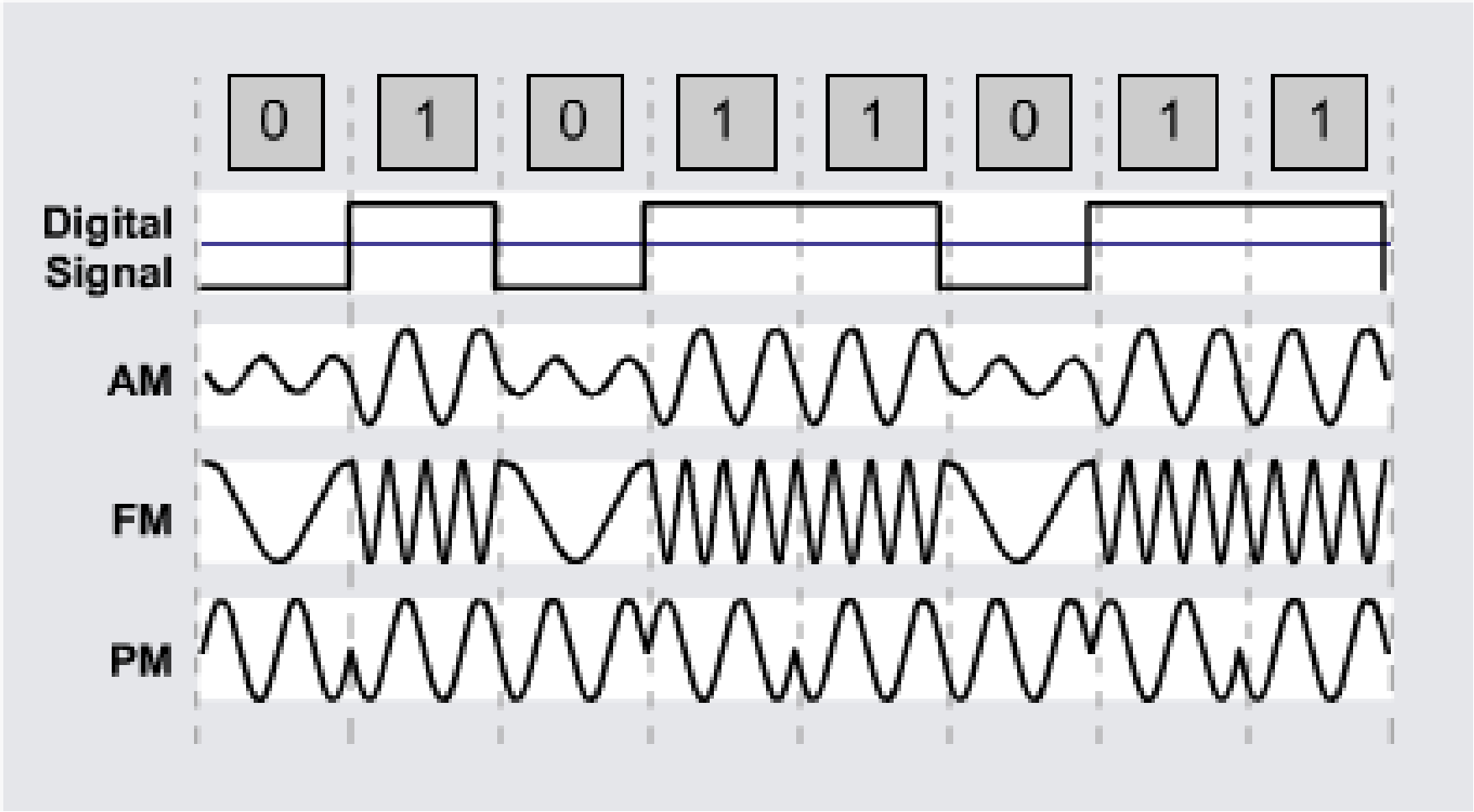


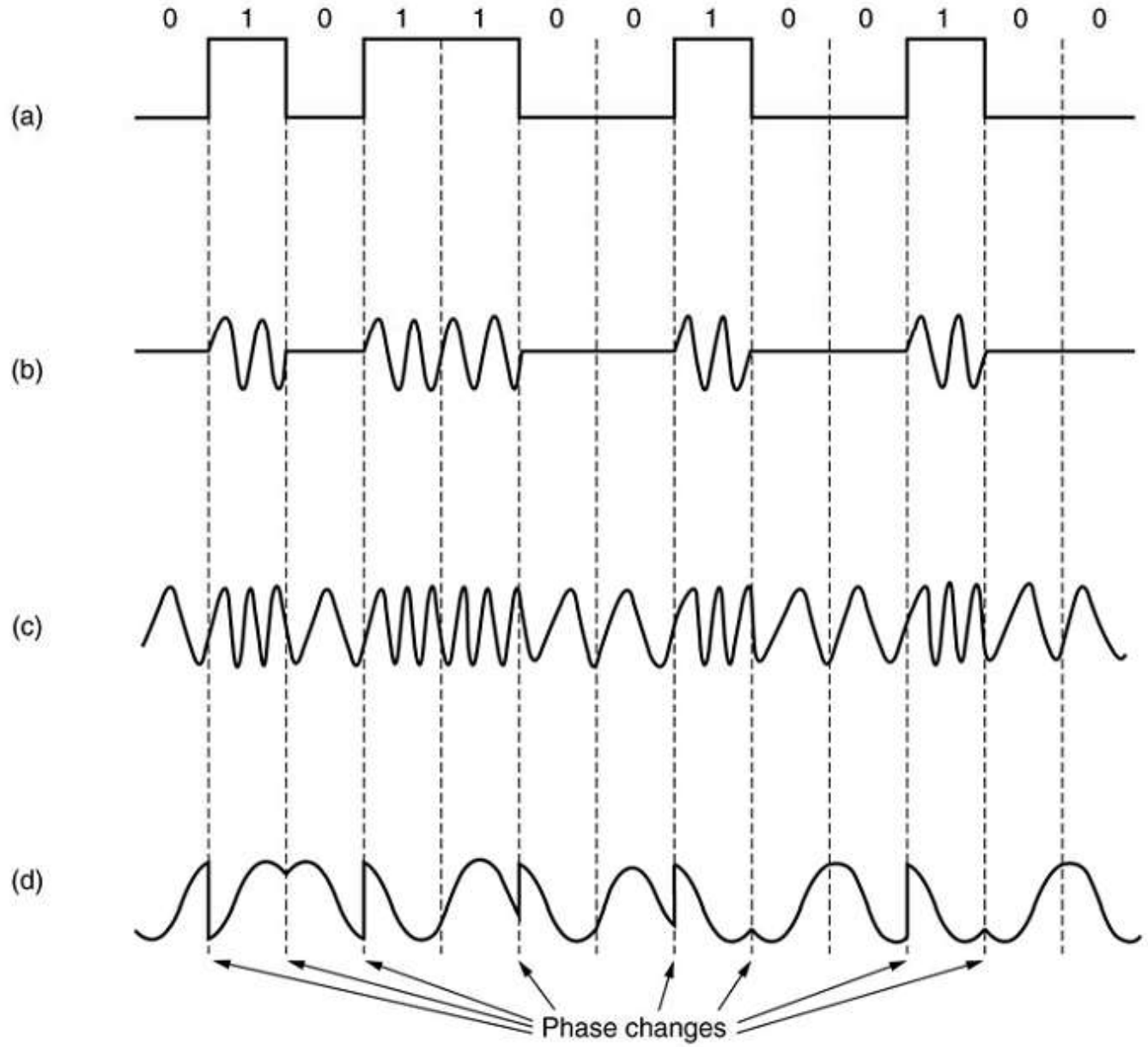
Şekil 1-5 : Kare Dalga



Şekil 1-2 : Verinin İletilmesi

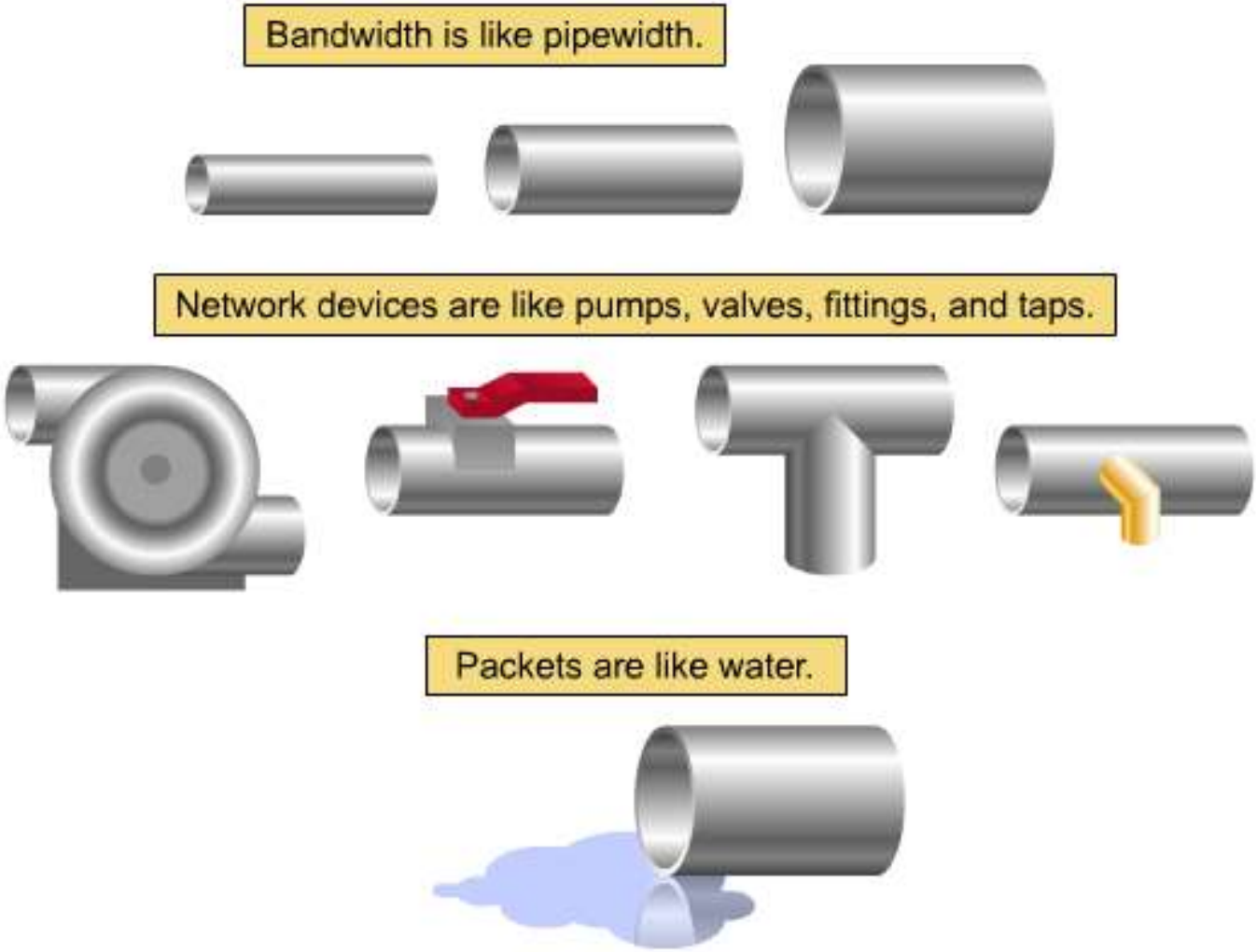
Modülasyon(Kipleme)





Şekil 1.6 Genlik (b), frekans (c) ve faz (d) modülasyonu ile sayısal verinin(a) ifade edilmesi. [1]'den alınmıştır.

Bant Genişliği



Otoban Benzetmesi

Bandwidth is like the number of lanes.



Network devices are like on-ramps, traffic signals, signs, and maps.



Packets are like vehicles.



3. Hafta

REFERANS MODELLERİ VE KATMANLAR (OSI, TCP)

Veri Aktarımı

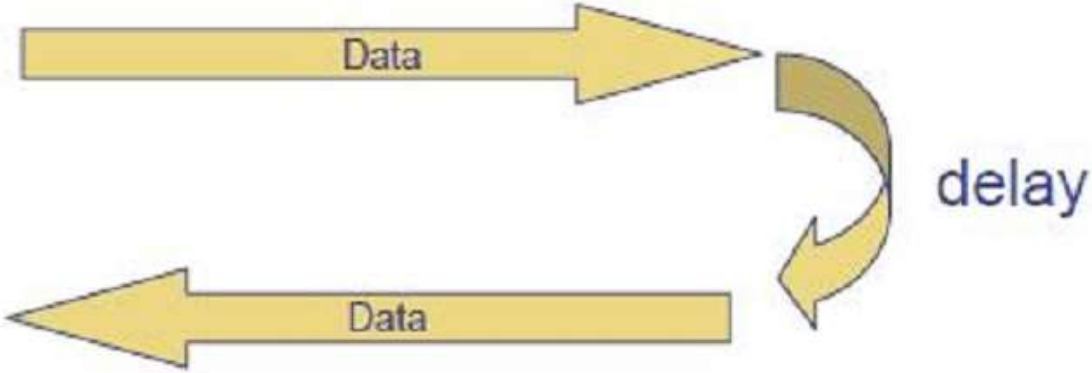
Simplex data flow

One direction, e.g. Television



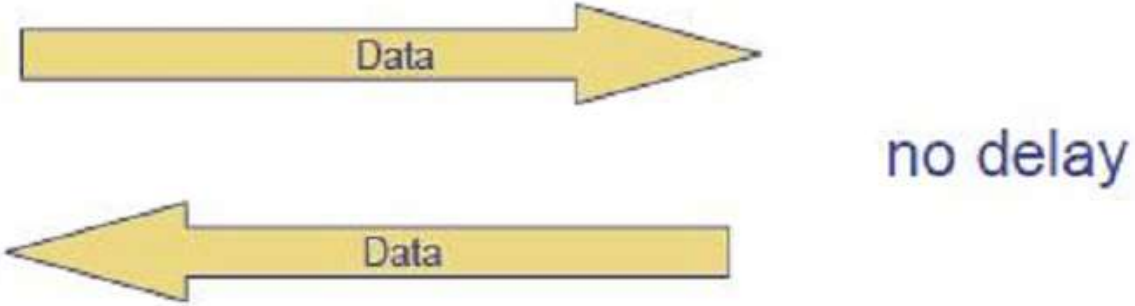
Half-Duplex data flow

Either direction, but only one way at a time, e.g. police radio

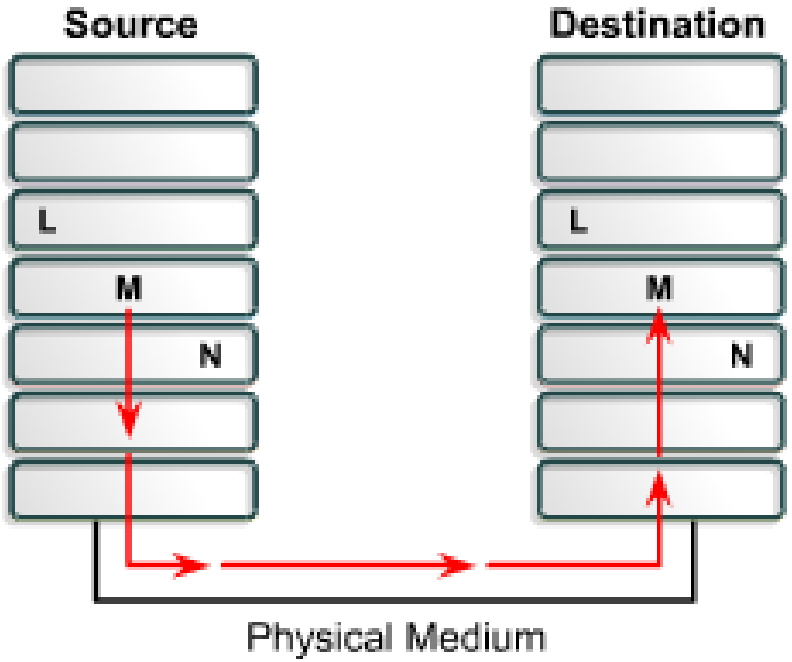


Full-Duplex data flow

Both directions at the same time e.g. telephone



Katmanlı Çalışma



L, M, N	Layers in our model of computer communications
Msource, Mdestination	Peer layers
→	Peer to peer communications
M layer Protocol	The rules by which Msource communicates with Mdestination

OSI Model

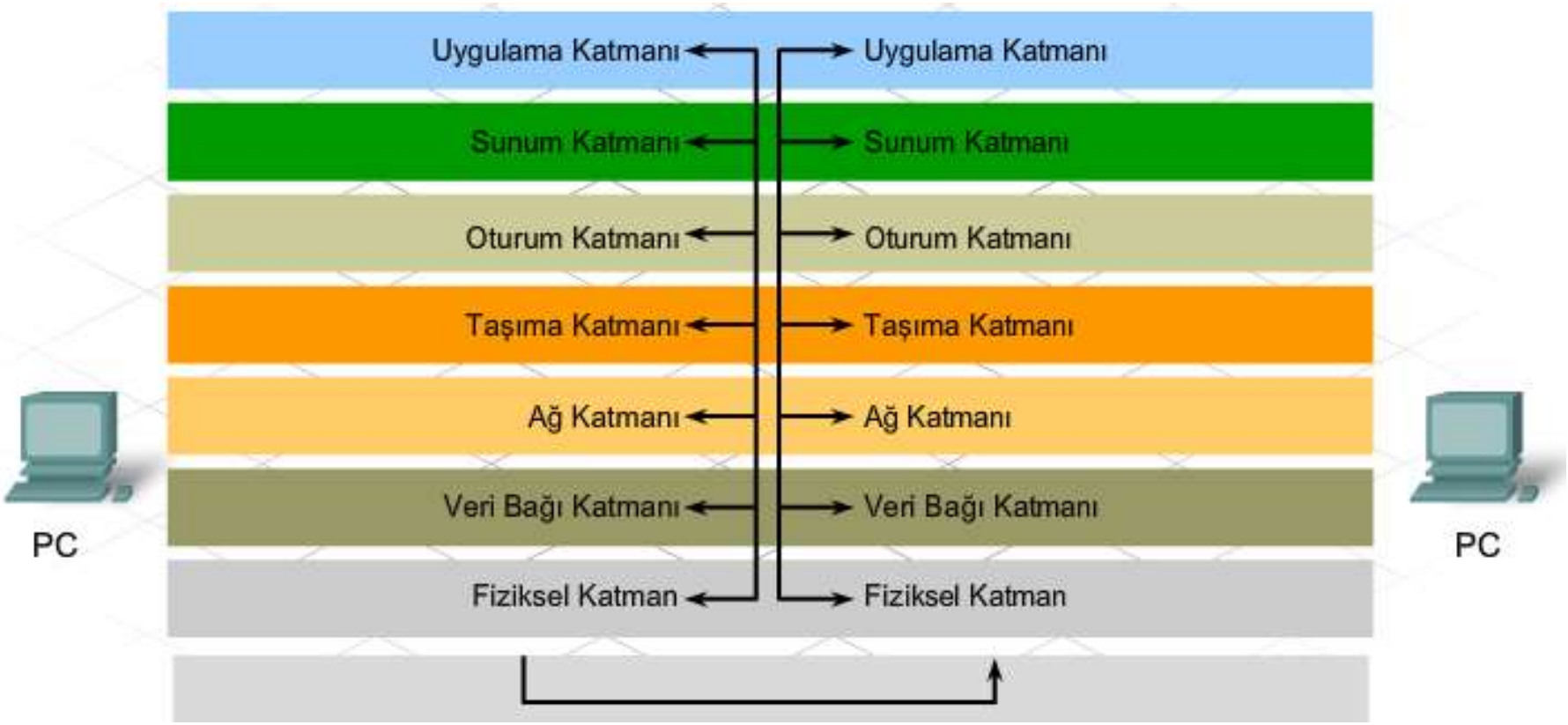
Open **S**ystems **I**nterconnection

(Açık Sistem Arabağlantısı Modeli)

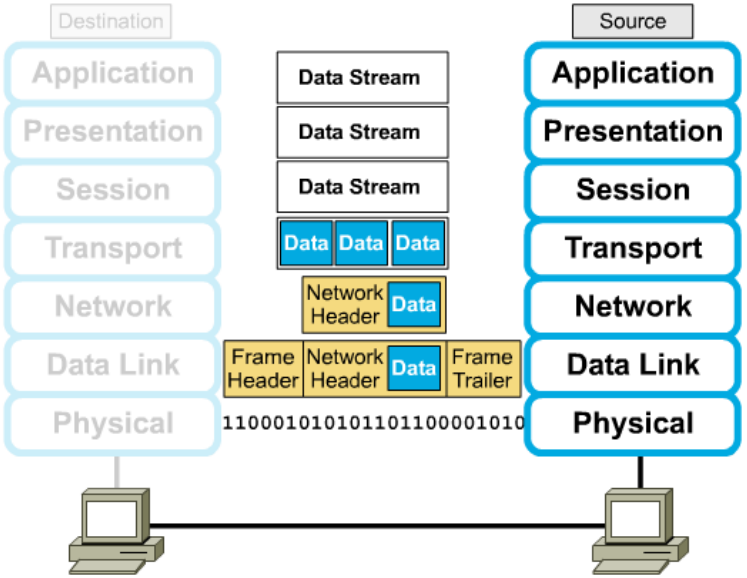
- ISO (International Organization for Standardization) (Uluslar arası Standartlar Kuruluşu) tarafından geliştirilmiştir.

OSI Modeli

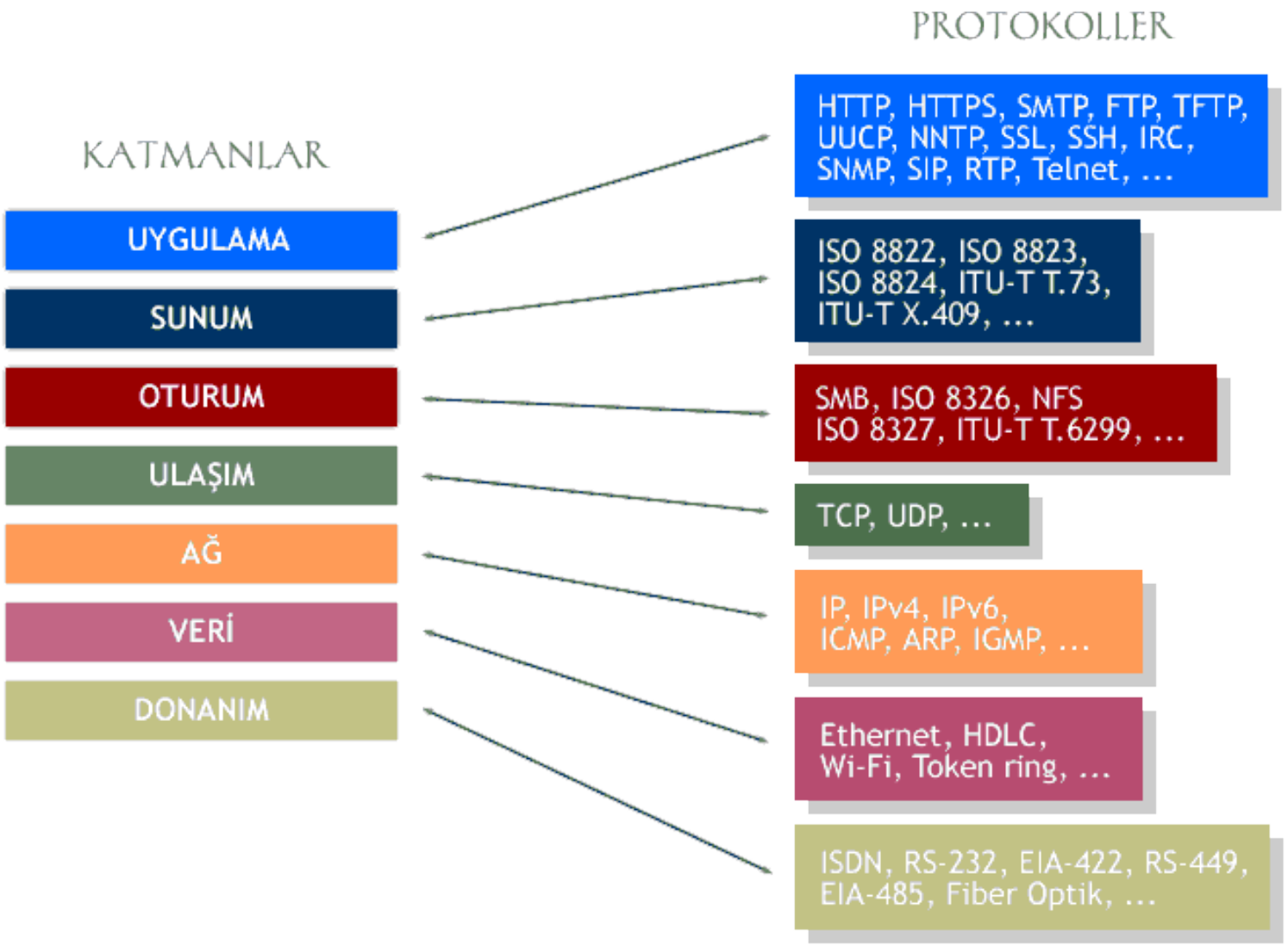
7	Uygulama Katmanı(Application Layer)	Uygulama katmanıdırlar, Genellikle Yazılım ile gerçeklenirler,En üst katman kullanıcıya en yakındır.
6	Sunum Katmanı(Presentation Layer)	
5	Oturum Katmanı (Session Layer)	
4	İletim Katmanı (Transport Layer)	Veri iletim işlemlerini gerçekler. Fiziksel ve Veri iletim katmanları yazılım veya donanım ile gerçeklenir.
3	Ağ Katmanı(Network Layer)	
2	Veri Bağlantı Katmanı(Data Link Layer)	
1	Fiziksel Katman(Physical Layer)	

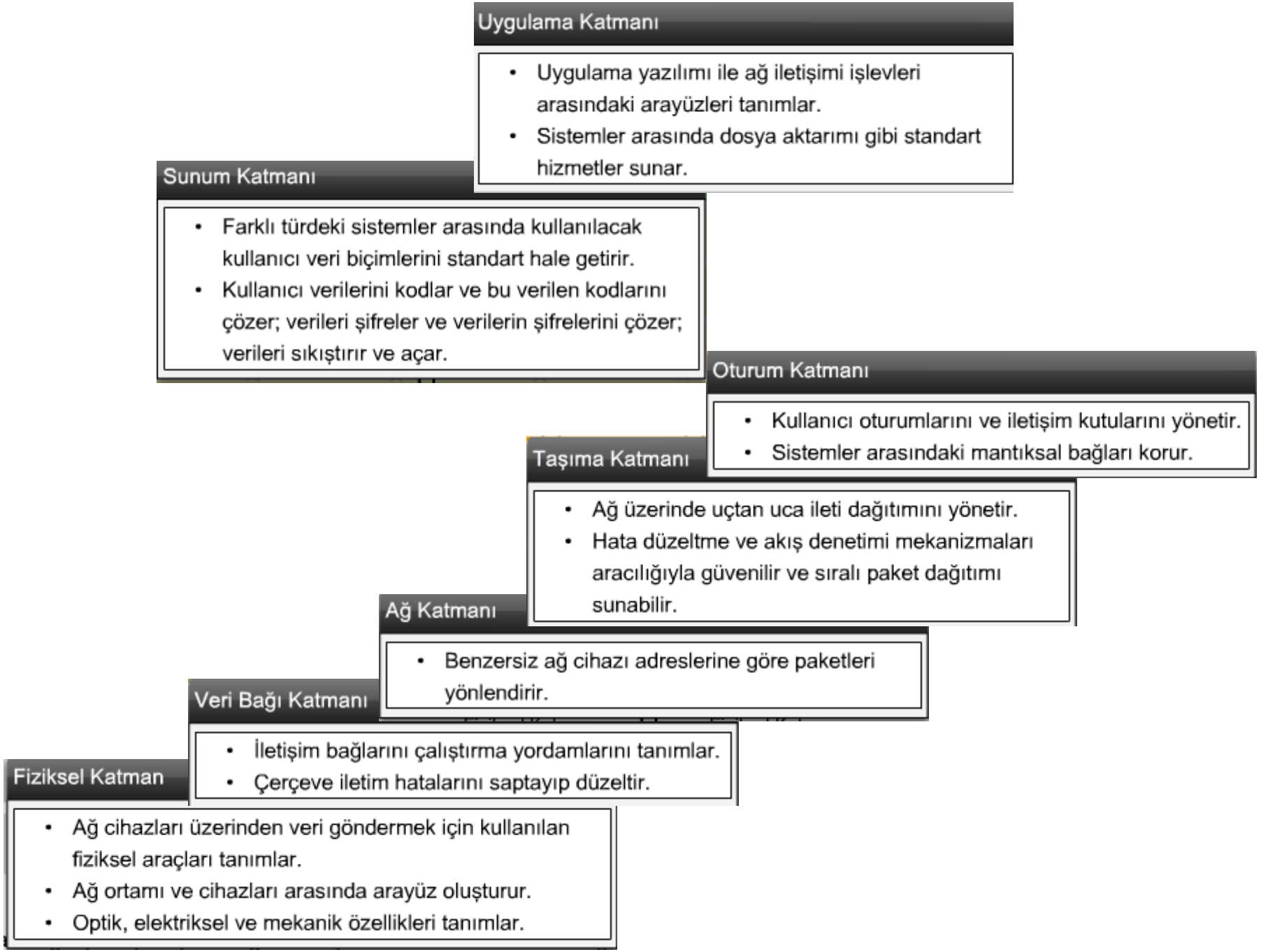

Uygulama Katmanı
Uygulama Katmanı
Sunum Katmanı
Sunum Katmanı
Oturum Katmanı
Oturum Katmanı
Taşıma Katmanı
Taşıma Katmanı
Ağ Katmanı
Ağ Katmanı
Veri Bağı Katmanı
Veri Bağı Katmanı
Fiziksel Katman
Fiziksel Katman
PC
PC


7 Application
6 Presentation
5 Session
4 Transport
3 Network
2 Data Link
1 Physical

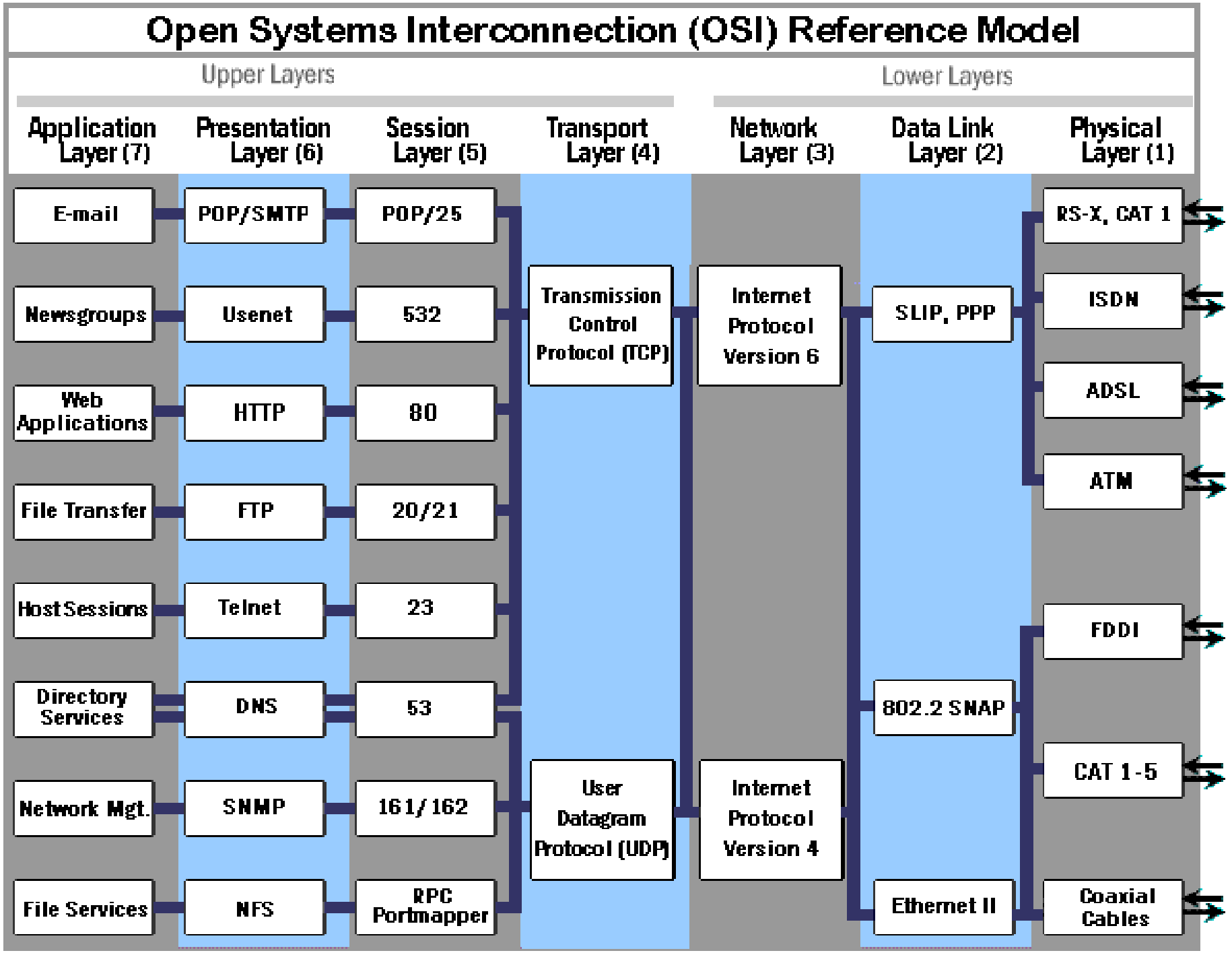

Destination
Source
Application
Presentation
Session
Transport
Network
Data Link
Physical
Data Stream
Data Stream
Data Stream
Data
Data
Data
Network Header
Data
Frame Header
Network Header
Data
Frame Trailer
1100010101011011000010101

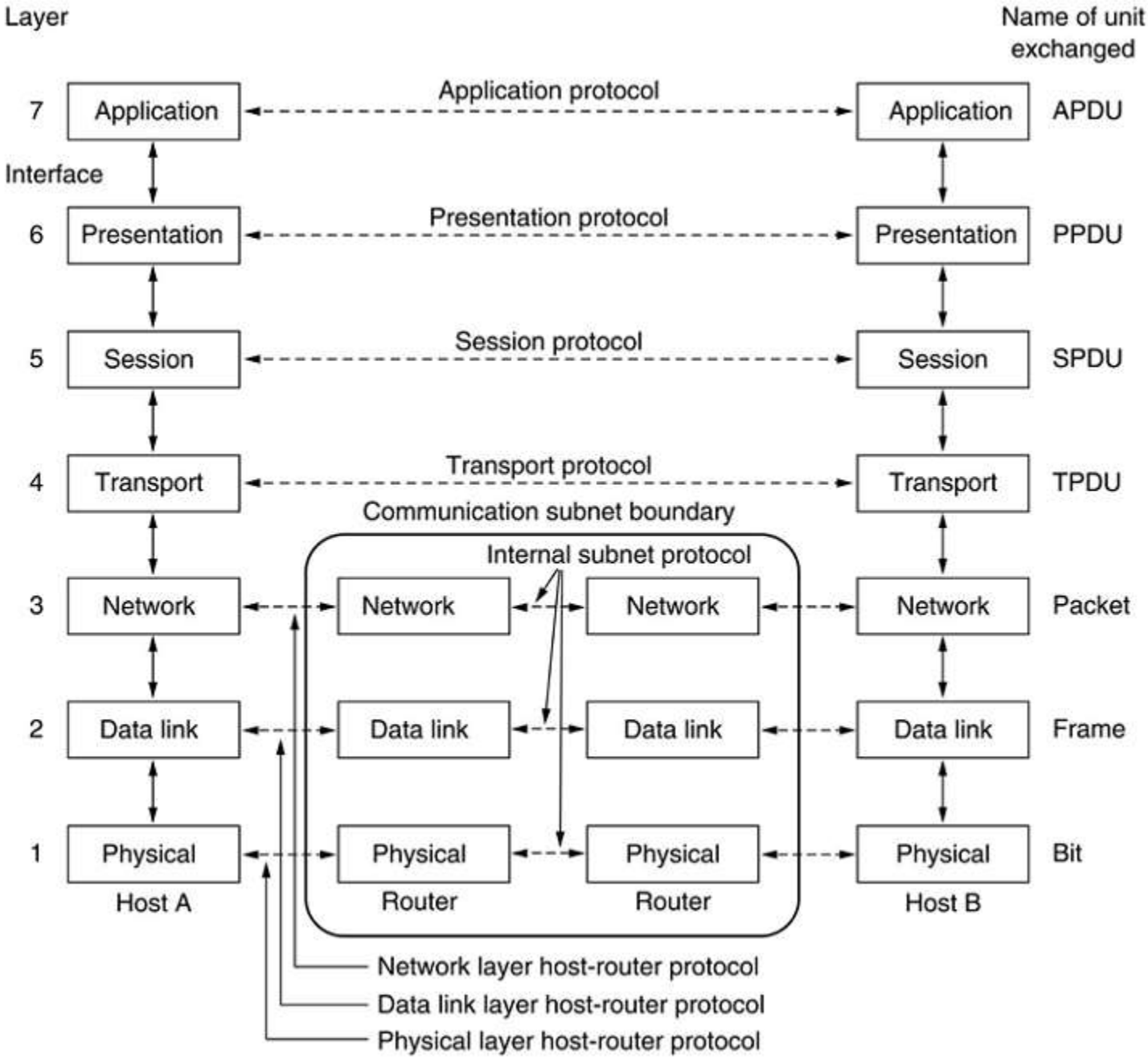
OSI Modeli


PROTOKOLLER
KATMANLAR
UYGULAMA
SUNUM
OTURUM
ULAŞIM
AĞ
VERİ
DONANIM
HTTP, HTTPS, SMTP, FTP, TFTP, UUCP, NNTP, SSL, SSH, IRC, SNMP, SIP, RTP, Telnet, ...
ISO 8822, ISO 8823, ISO 8824, ITU-T T.73, ITU-T X.409, ...
SMB, ISO 8326, NFS ISO 8327, ITU-T T.6299, ...
TCP, UDP, ...
IP, IPv4, IPv6, ICMP, ARP, IGMP, ...
Ethernet, HDLC, Wi-Fi, Token ring, ...
ISDN, RS-232, EIA-422, RS-449, EIA-485, Fiber Optik, ...


Uygulama Katmanı
• Uygulama yazılımı ile ağ iletişimi işlevleri arasındaki arayüzleri tanımlar.
• Sistemler arasında dosya aktarımı gibi standart hizmetler sunar.
Sunum Katmanı
• Farklı türdeki sistemler arasında kullanılacak kullanıcı veri biçimlerini standart hale getirir.
• Kullanıcı verilerini kodlar ve bu verilen kodlarını çözer; verileri şifreler ve verilerin şifrelerini çözer; verileri sıkıştırır ve açar.
Oturum Katmanı
• Kullanıcı oturumlarını ve iletişim kutularını yönetir.
• Sistemler arasındaki mantıksal bağları korur.
Taşıma Katmanı
• Ağ üzerinde uçtan uca ileti dağıtımını yönetir.
• Hata düzeltme ve akış denetimi mekanizmaları aracılığıyla güvenilir ve sıralı paket dağıtımı sunabilir.
Ağ Katmanı
• Benzersiz ağ cihazı adreslerine göre paketleri yönlendirir.
Veri Bağı Katmanı
• İletişim bağlarını çalıştırma yordamlarını tanımlar.
• Çerçeve iletim hatalarını saptayıp düzeltir.
Fiziksel Katman
• Ağ cihazları üzerinden veri göndermek için kullanılan fiziksel araçları tanımlar.
• Ağ ortamı ve cihazları arasında arayüz oluşturur.
• Optik, elektriksel ve mekanik özellikleri tanımlar.

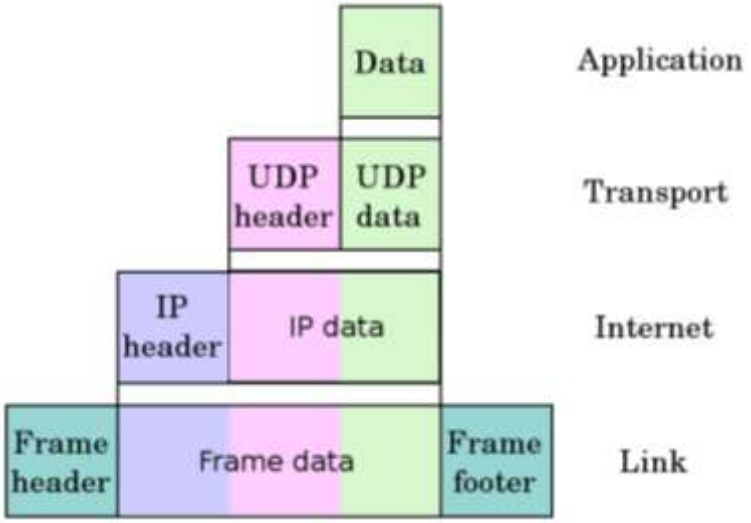
OSI Modeli	Katman	Açıklama
Uygulama	7	Uygulamalara ağ hizmetleri sağlamakla yükümlüdür
Sunum	6	Uygulama katmanı için standart bir arayüz sağlamak üzere veri biçimlerini dönüştürür
Oturum	5	Yerel ve uzak uygulama arasındaki bağlantıları oluşturur, yönetir ve sonlandırır.
Taşıma	4	Ağ üzerinde güvenilir taşıma ve akış denetimi sağlar
Ağ	3	Mantıksal adreslemeden ve yönlendirme etki alanından sorumludur
Veri Bağı	2	Fiziksel adresleme ve ortam erişim yordamları sağlar
Fiziksel	1	Aygıtlar için tüm elektriksel ve fiziksel özellikleri tanımlar

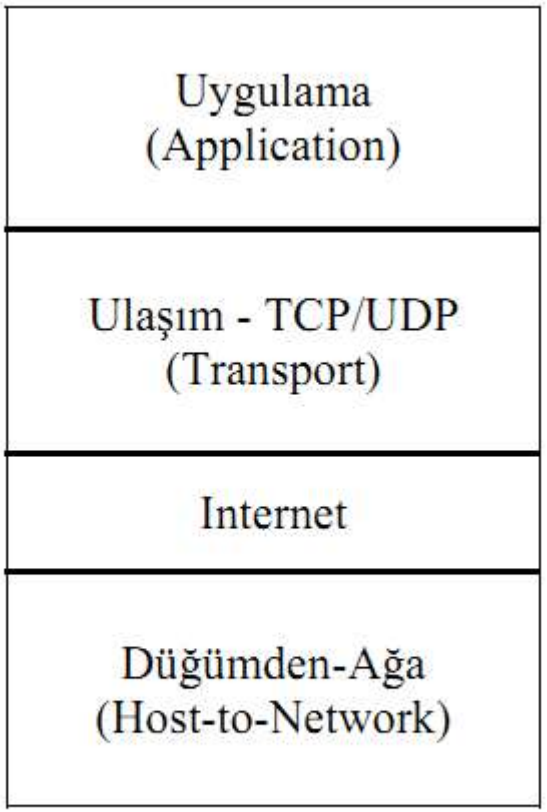

Open Systems Interconnection (OSI) Reference Model
Upper Layers
Lower Layers
Application Layer (7)
Presentation Layer (6)
Session Layer (5)
Transport Layer (4)
Network Layer (3)
Data Link Layer (2)
Physical Layer (1)
E-mail
POP/SMTP
POP/25
Newsgroups
Usenet
532
Web Applications
HTTP
80
File Transfer
FTP
20/21
Host Sessions
Telnet
23
Directory Services
DNS
53
Network Mgt.
SNMP
161/162
File Services
NFS
RPC Portmapper
Transmission Control Protocol (TCP)
User Datagram Protocol (UDP)
Internet Protocol Version 6
Internet Protocol Version 4
SLIP, PPP
802.2 SNAP
Ethernet II
RS-X, CAT 1
ISDN
ADSL
ATM
FDDI
CAT 1-5
Coaxial Cables


Layer
Name of unit exchanged
7
Application
Application protocol
Application
APDU
Interface
6
Presentation
Presentation protocol
Presentation
PPDU
5
Session
Session protocol
Session
SPDU
4
Transport
Transport protocol
Transport
TPDU
Communication subnet boundary
Internal subnet protocol
3
Network
Network
Network
Network
Packet
2
Data link
Data link
Data link
Data link
Frame
1
Physical
Physical
Physical
Physical
Bit
Host A
Router
Router
Host B
Network layer host-router protocol
Data link layer host-router protocol
Physical layer host-router protocol

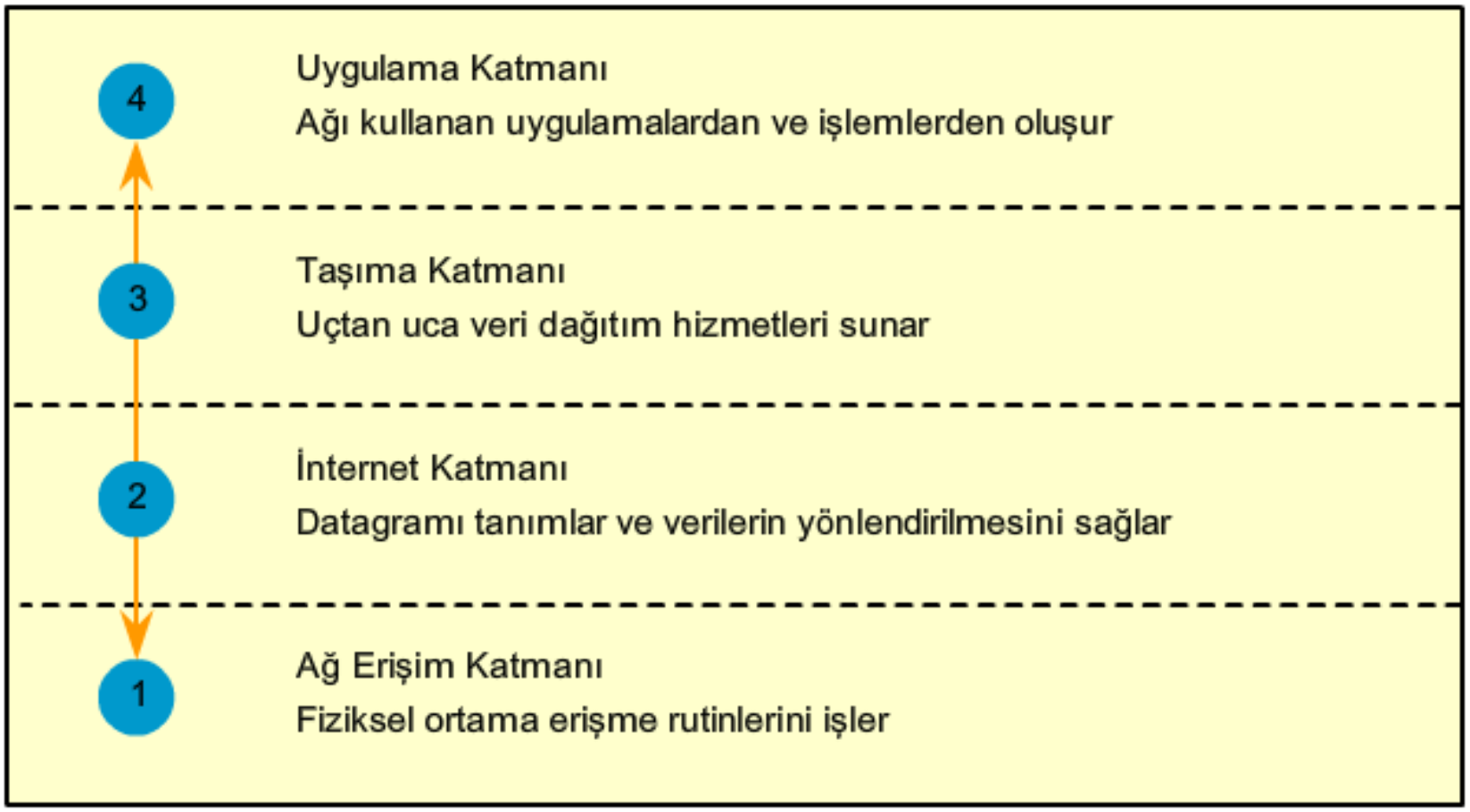
TCP/IP Model

- TCP/IP Modeli Amerikan Savunma Bakanlığı tarafından heterojen ağlarda kesintisiz bağlantılı iletişim için geliştirilmiştir.
- 4 katmandan oluşur.

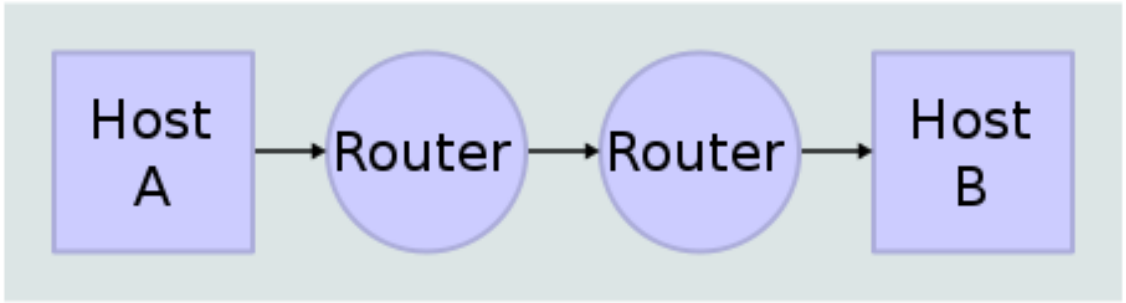




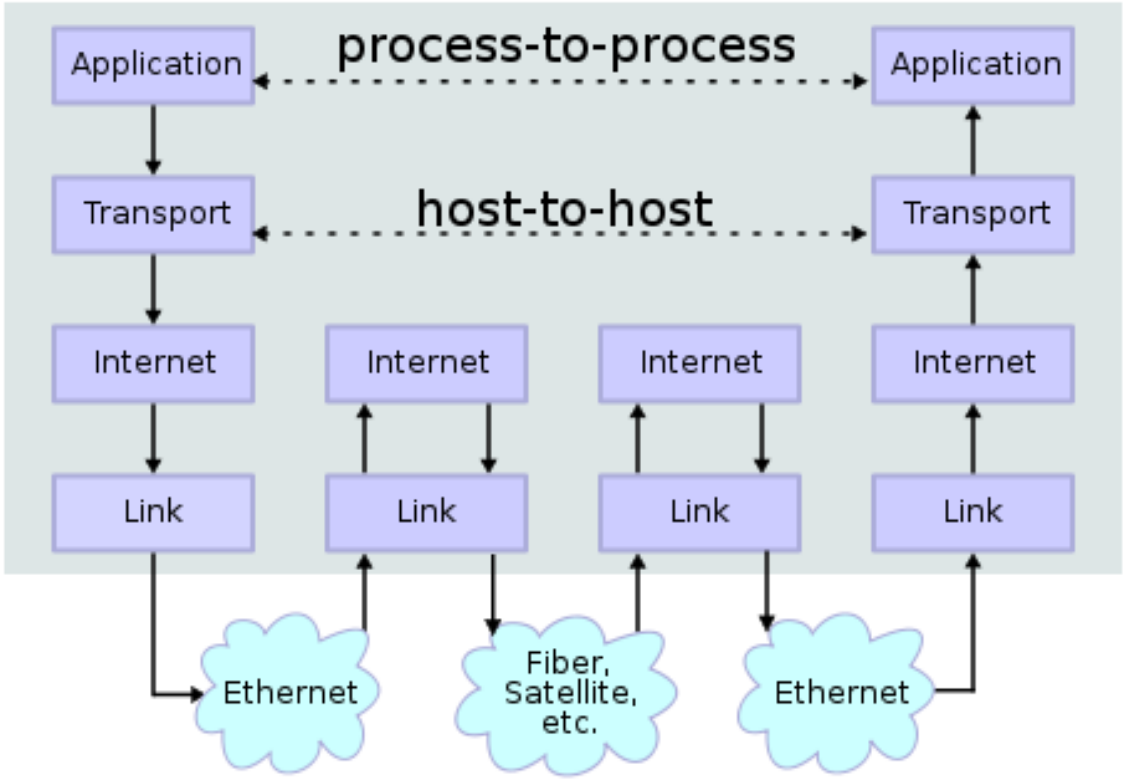
Şekil 8.4 TCP/IP Referans modeli


4
Uygulama Katmanı
Ağı kullanan uygulamalardan ve işlemlerden oluşur
3
Taşıma Katmanı
Uçtan uca veri dağıtım hizmetleri sunar
2
İnternet Katmanı
Datagramı tanımlar ve verilerin yönlendirilmesini sağlar
1
Ağ Erişim Katmanı
Fiziksel ortama erişme rutinlerini işler

Network Topology



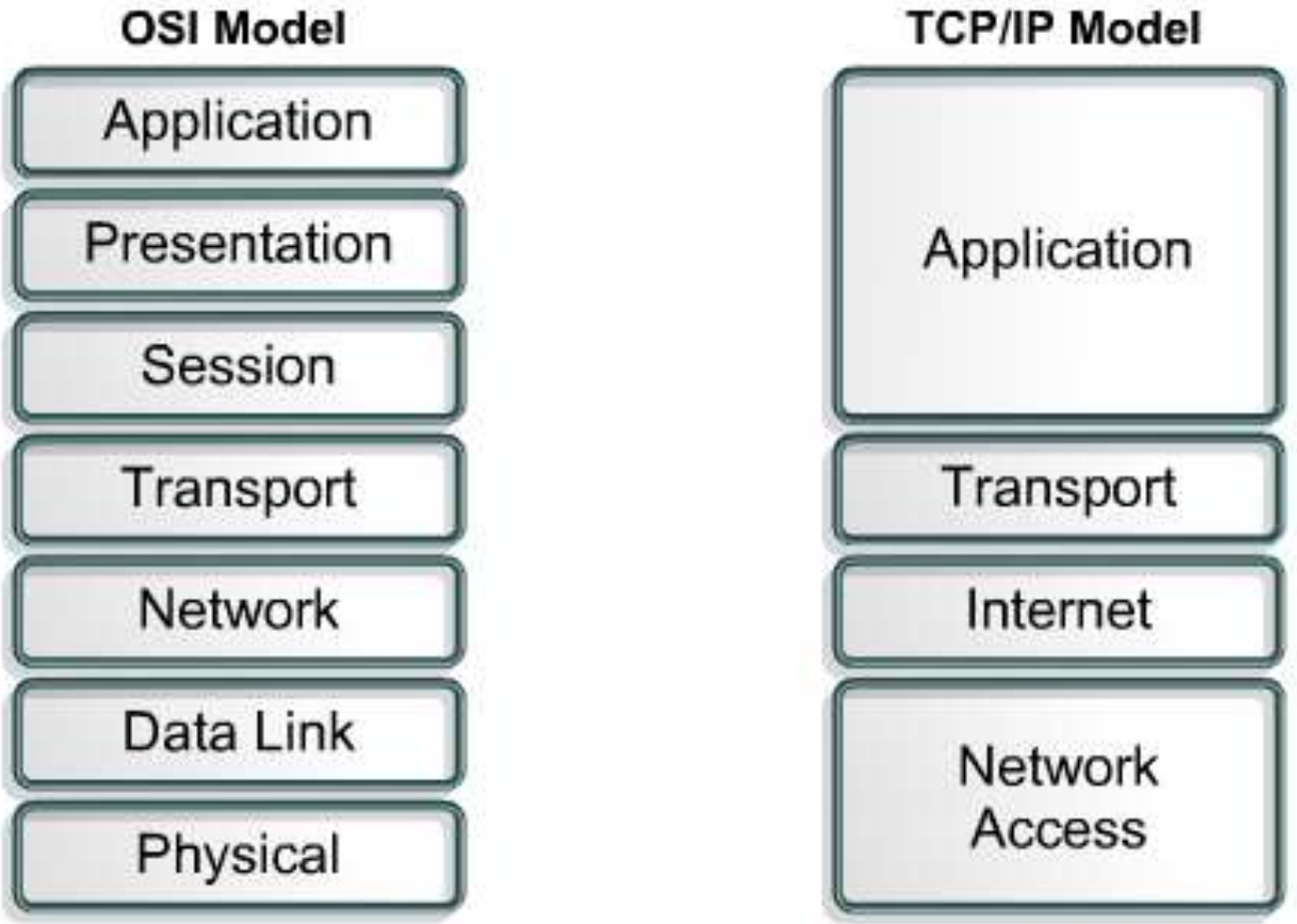
Data Flow



TCP/IP

Katman	Açıklama	Protokoller
Uygulama	TCP/IP uygulama protokollerini ve ana bilgisayar programlarının ağı kullanmak için taşıma katmanı hizmetleriyle nasıl bir arabirim oluşturacağını tanımlar.	HTTP, Telnet, FTP, TFTP, SNMP, DNS, SMTP, X Windows, diğer uygulama protokolleri
Taşıma	Ana bilgisayarlar arasında iletişim oturumu yönetimi sağlar. Veri taşınırken kullanılan bağlantının hizmet düzeyini ve durumunu tanımlar.	TCP, UDP, RTP
Internet	Verileri IP veri birimleri olarak paketler. Bu paketler, veri birimlerini ana bilgisayarlar ve ağlar arasında iletmek için kullanılan kaynak ve hedef bilgilerini içerir. IP veri birimlerinin yönlendirilmesini gerçekleştirir.	IP, ICMP, ARP, RARP
Ağ arabirimi	Koaksiyel kablo, optik fiber veya çift bükümlü bakır kablo gibi bir ağ ortamıyla doğrudan arabirim oluşturan donanım aygıtları tarafından bitlerin elektriksel olarak nasıl işaret haline getirileceği de dahil olmak üzere verilerin fiziksel olarak ağ içinden nasıl gönderileceğini belirtir.	Ethernet, Token Ring, FDDI, X.25, Frame Relay, RS-232, v.35

OSI ve TCP/IP


OSI Model
Application
Presentation
Session
Transport
Network
Data Link
Physical
TCP/IP Model
Application
Transport
Internet
Network Access

TCP/IP Modeli	OSI Modeli
Uygulama	Uygulama Sunum Oturum
Taşıma	Taşıma
İnternet	Ağ
Ağ Erişimi	Veri Bağı Fiziksel

4. Hafta

AĞ BİLEŞENLERİ, AĞ SINIFLANDIRMASI, TOPOLOJİLER

Ağ Bileşenleri

- Ağ Sınıflandırılması
  - Lan
  - Man
  - Wan
- Ağ Cihazları
  - Aktif
  - Pasif
- Topoloji



Ağın Sınıflandırılması

- Coğrafi koşullara göre;
  - LAN, MAN, WAN
- Topolojilerine göre;
  - Bus, Ring, Star, Tree, Mesh
- Ortamlarına göre;
  - OSI, TCP/IP
  - Ethernet, Token Ring, FDDI, ATM
- İletim Yöntemleri;
  - Aktif (Ağ Cihazları);
    - Modem, NIC, Repeater, Hub, Switch, Router
  - Pasif (Kablolar);
    - Coaxial, UTP, STP, Fiber

LAN, MAN, WAN

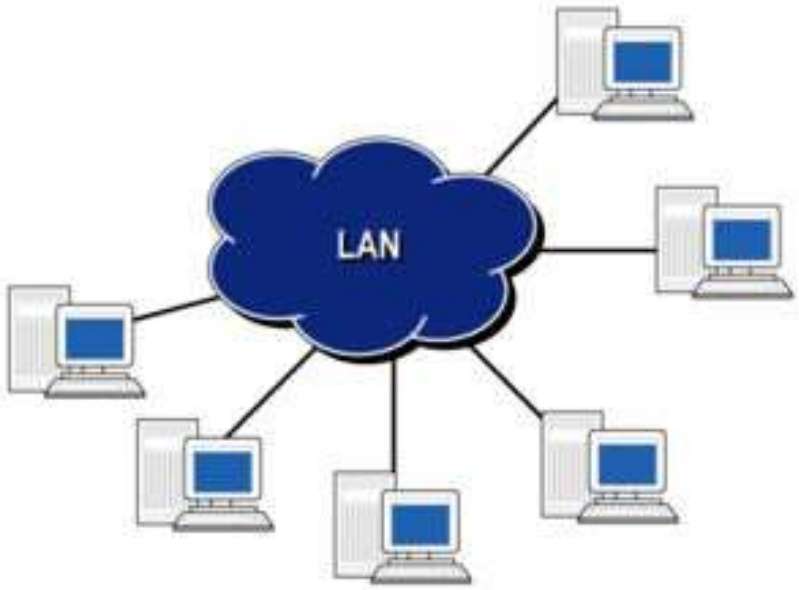
- LAN - Yerel alan ağları
  (Local Area Networks, LANs)

- MAN - Metropol alan ağları
  (Metropolitan Area Networks, MAN)

- WAN - Geniş alan ağları
  (Wide Area Networks, WAN)

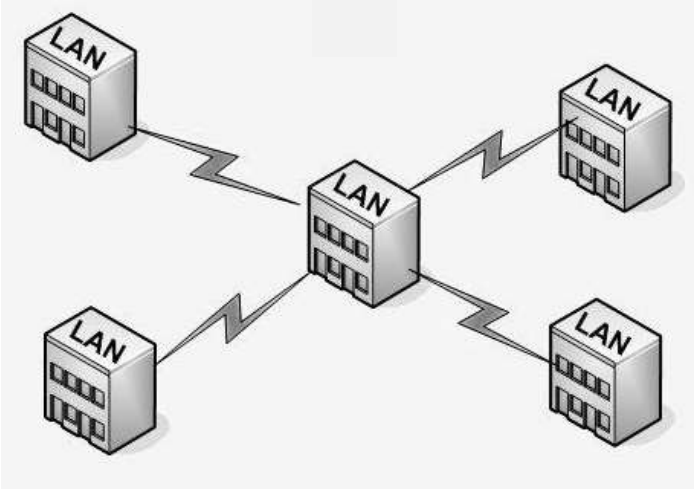
LAN

- Ev, okul, laboratuvar, iş binaları vb. gibi sınırlı coğrafi alanda bilgisayarları ve araçları birbirine bağlayan bir bilgisayar ağıdır.



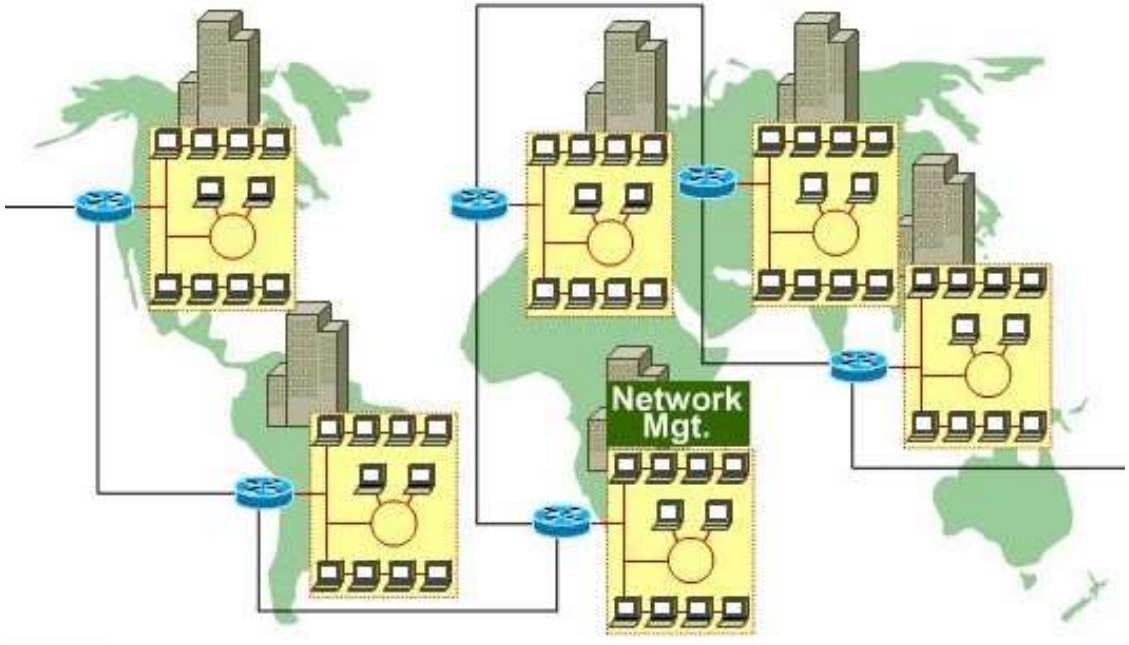
MAN

- LAN ağlarından daha büyük bir ağ yapısıdır.
- Metropol olarak adlandırılmasının sebebi genellikle şehrin bir kısmını ya da tamamını kapsamasındandır.
- Kampüs ağları adıyla da anılan bu ağlar, üniversite kampüslerinde ve büyük işyerlerinde kullanılır.
- Değişik donanım ve aktarım ortamları kullanılır.



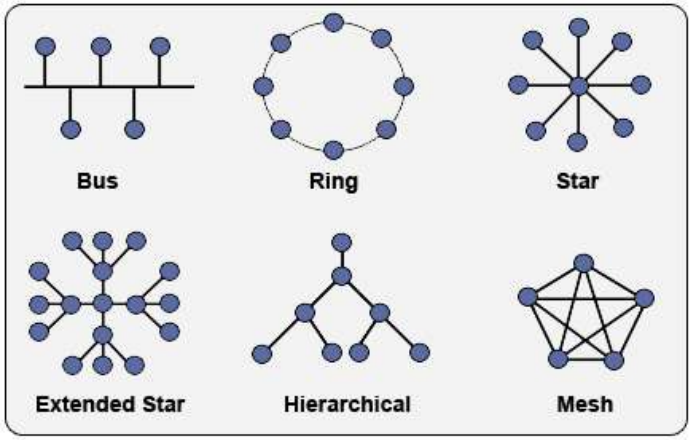
WAN

- Birden fazla cihazın birbiri ile iletişim kurmasını sağlayan fiziksel veya mantıksal büyük ağdır.
- Yerel alan ağlarının birbirine bağlanmasını sağlayan çok geniş ağlardır. En geniş alan ağı İnternettir.

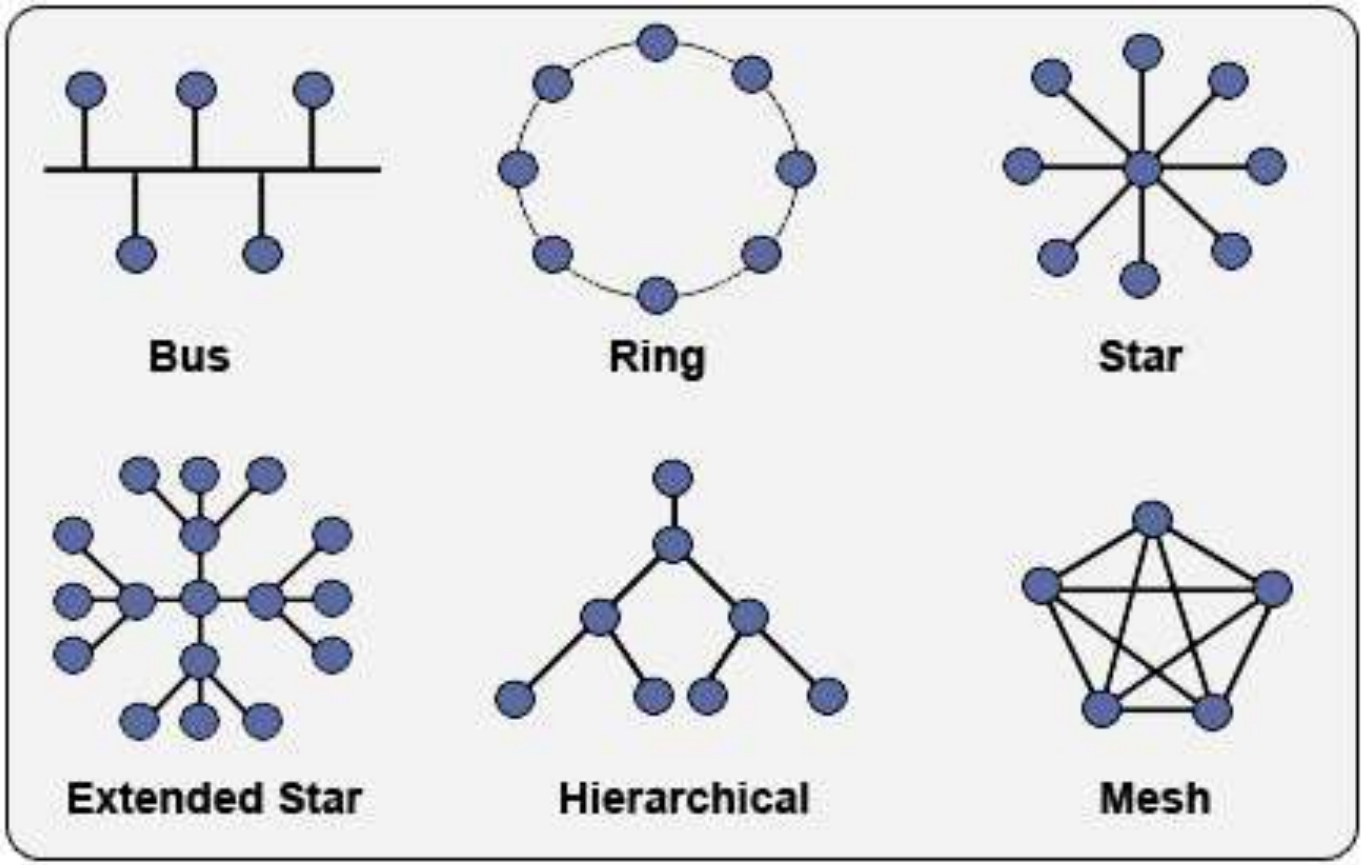


Ağ Topolojileri

- Bir ağdaki bilgisayarların nasıl yerleşeceğini, nasıl bağlanacağını, veri iletiminin nasıl olacağını belirleyen genel yapıdır.
  - Fiziksel topoloji: Ağın fiziksel olarak nasıl görüneceğini belirler (Fiziksel katman)
  - Mantıksal topoloji: Bir ağdaki veri akışının nasıl olacağını belirler (Veri iletim katmanı)



Topolojiler



Topolojiler ve Kablolar

Topoloji	Bus		Ring		Yıldız		Mesh
Kullanım	-		Kısmen		Birincil		Kablosuz
Hub & Swich	-		-		Kullanır		-
Yöntem	**Linear Bus**		**Token Ring**		**Ethernet**	**Fiber Optik**	**Ad-Hoc**
Kablolar							
Coaxial	**Kullanır**		İlk Evrelerinde...			-	-
STP / FSTP	-		**Kullanır**			-	-
UTP	-		Kısmen		**Kullanır**	-	-
Fiber Optik	-		-		-	**Kullanır**	-
Bağlayıcılar							
BNC	Coaxial (Thinnet: RG58 / Thicknet: RG8)					-	-
RJ45	-		Sarmal Çiftli (STP/UTP/FSTP)			-	-
SC/ST	-		-		-	Fiber Optik	-
Limitler	RG58	RG8	STP	UTP			
PC (adet)	30	100	260	72	1.024 / Switch & Hub Başına...		-
Uzunluk (m)	185	500	100 / 2 Birim Arasında...			2.000 / 2 Birim...	-

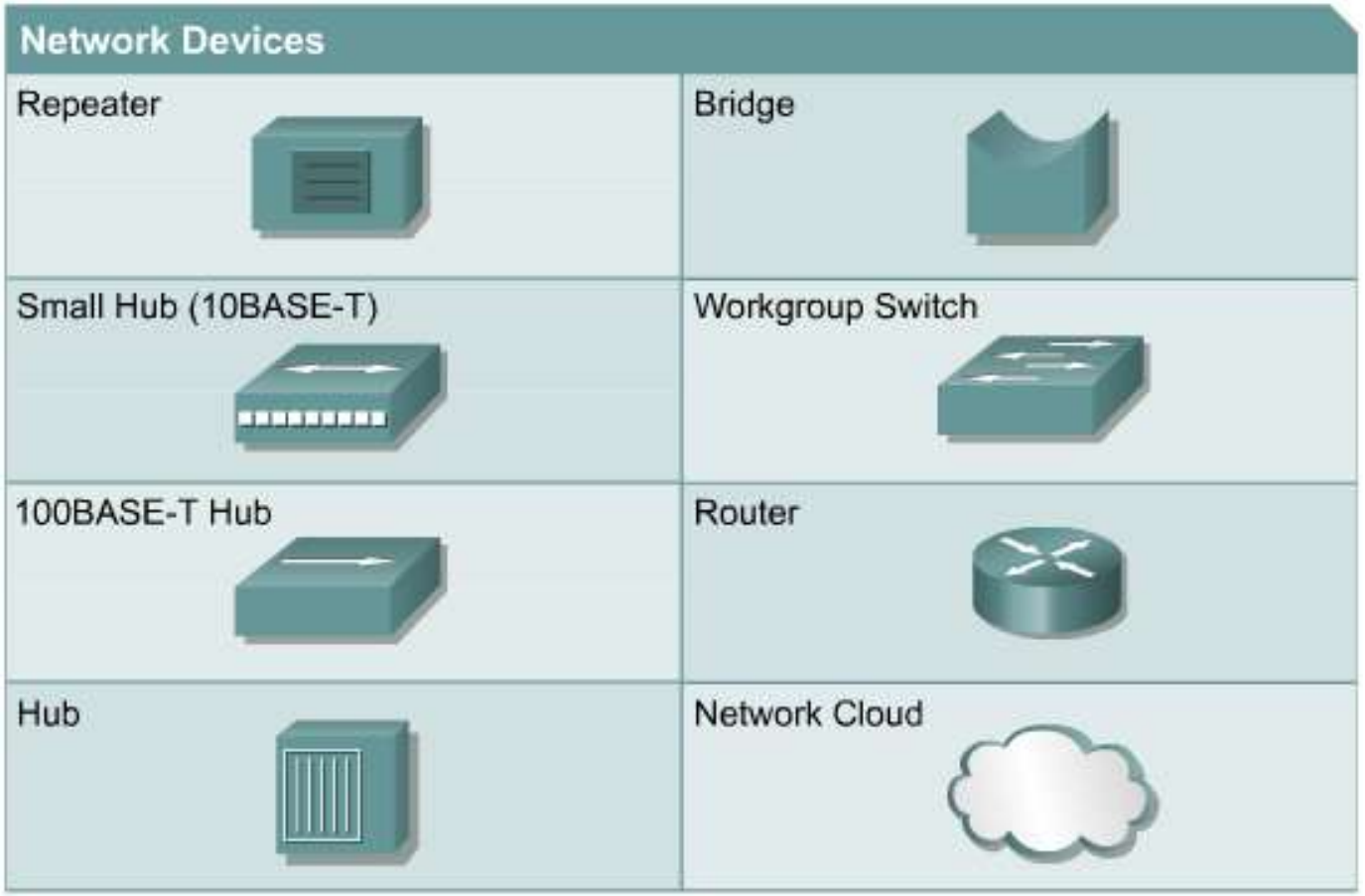
5. Hafta

AĞ DONANIMLARI, KABLOLAMA

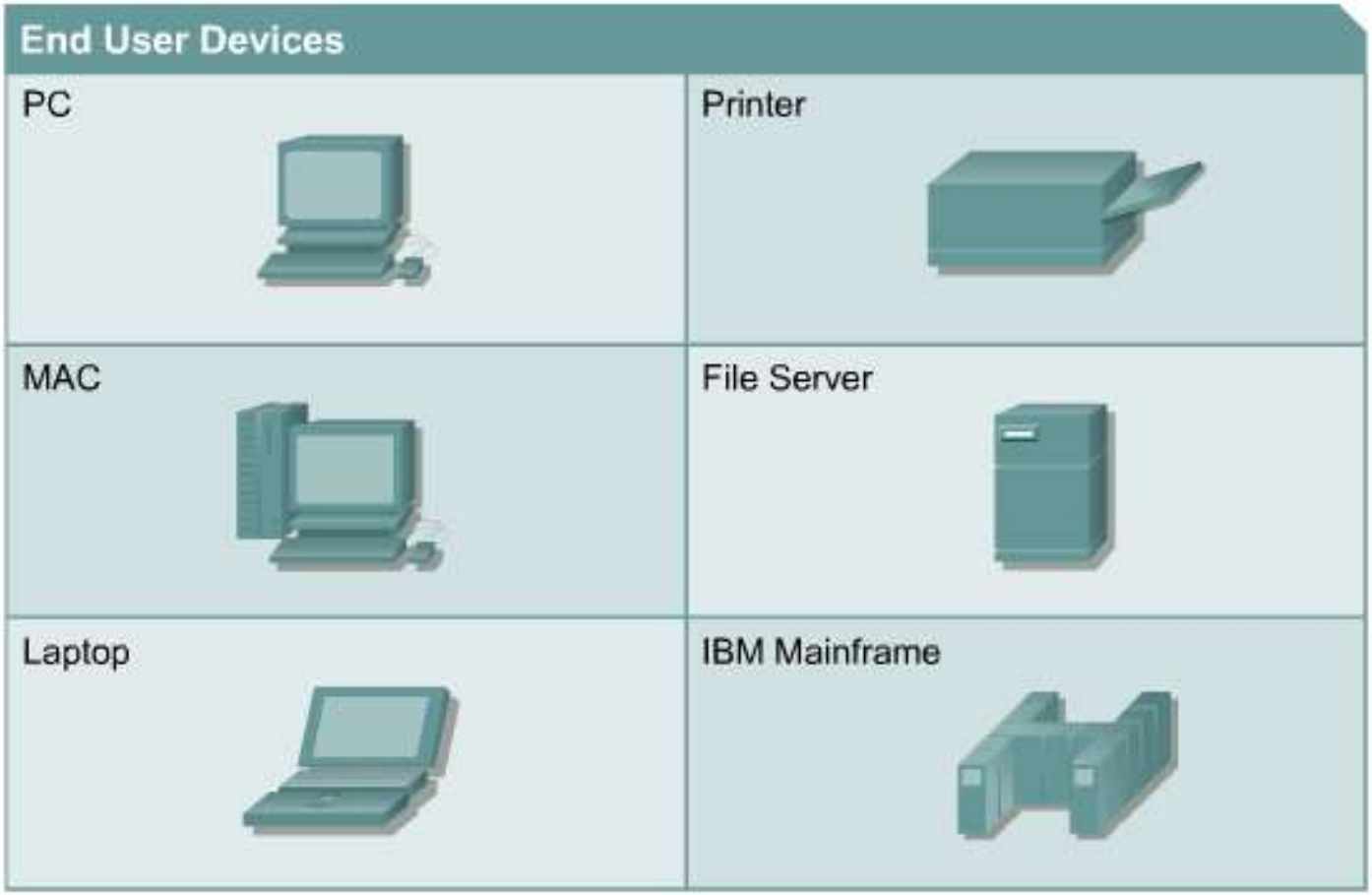
Ağ Donanımları(Cihazları)

- NIC – Network Interface Card(Ağ Arabirim Kartı)
- Repeater (Tekrarlayıcı-Yenileyici)
- Hub (Dağıtıcı)
- Switch (Anahtar)
- Bridge (Köprü)
  - Brouter (Köprü-Yönlendirici)
- Router (Yönlendirici)
- Gateway (Ağ Geçidi - Geçityolu)
- Firewall (Ateş Duvarı)

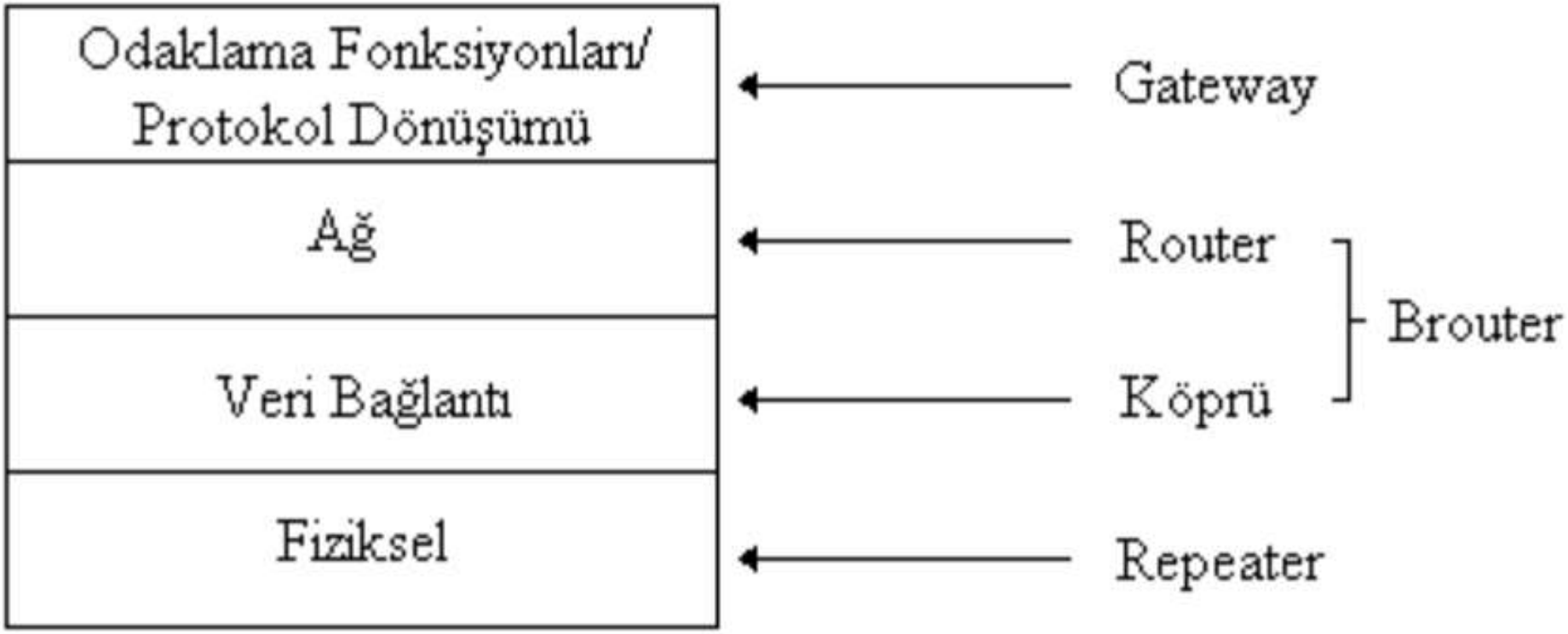
Ağ Cihazları ve Simgeleri



Ağ Cihazları ve Simgeleri



Katmanlara göre cihazlar



OSI ve Ağ Cihazları

OSI Katmanı	**Cihaz**
Uygulama	Ağ geçidi (Gateway)
Sunum	Ağ geçidi (Gateway)
Oturum	Ağ geçidi (Gateway)
Taşıma	Ağ geçidi (Gateway)
Ağ	Yönlendirici (Router) Katman 3 Switch
Veri İletim	Köprü (Bridge) Katman 2 Switch
Fiziksel	NIC, Yineleyici (Repeater) Hub Kablo, Alıcı ve verici

NIC – Network Interface Card (Ağ Arabirim Kartı)

- Ağ adaptörü veya ağ kartı (ethernet) kartı olarak adlandırılır.
- Sinyalleri alma gönderme işlemlerini yapar.
- Veri paketlerini parçalara ayırma birleştirme işlemlerini yapar.
- Fiziksel adrese sahiptir (MAC Adresi) 48 bittir. (16 lık)
- OSI de Fiziksel(1) ve Veri iletim(2) katmanlarında yer alır.
- Ethernet, ATM, FDDI, TokenRing, ISDN kullanılan teknolojilerdir.
- PCI, USB, PCMCIA bağlantı yuvalarına takılırlar.

NIC







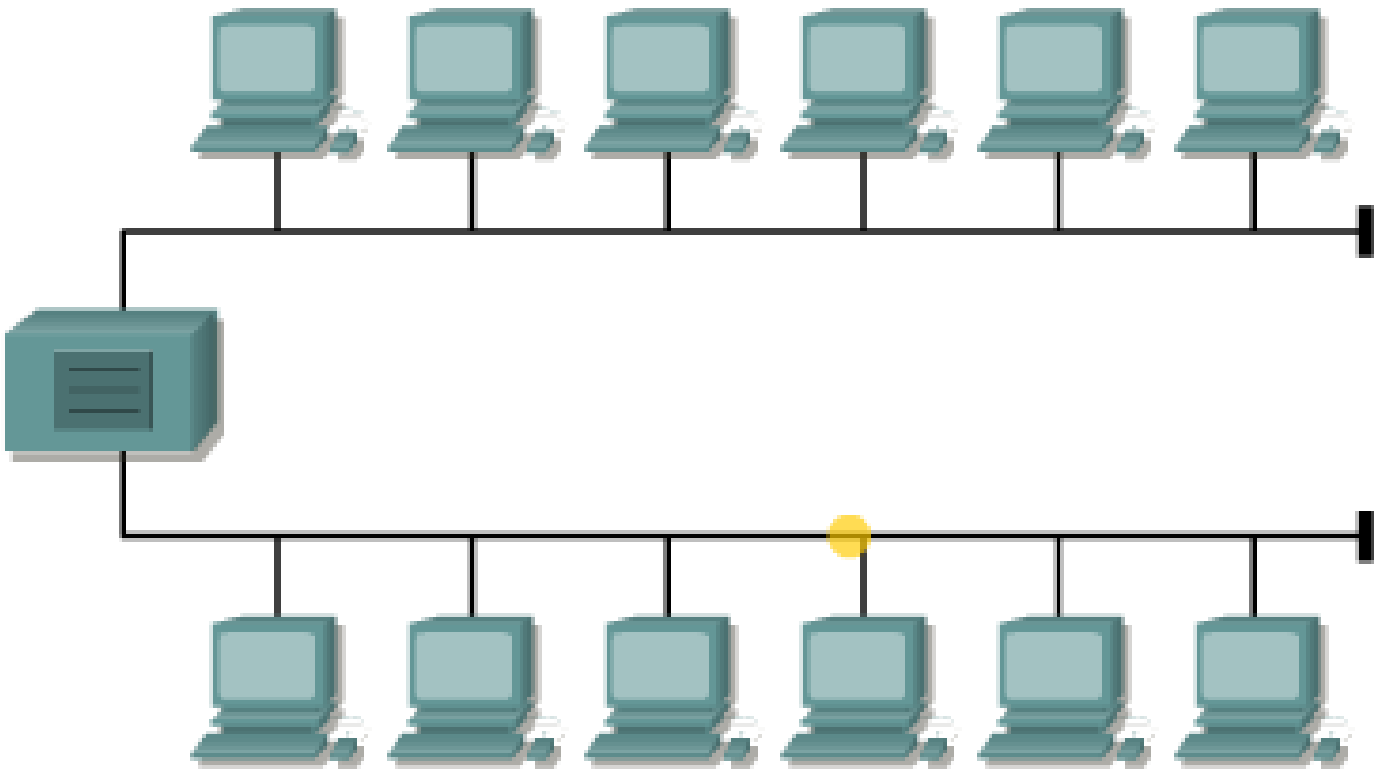




Repeater(Tekrarlayıcı-Yineleyici)

- Ağın genişletilmesinde kullanılır.
- Sinyallerin daha uzun mesafelere ulaştırılmasını sağlar.
- Farklı kablo türlerinde farklı mesafelerde kullanılır.
- Verileri sadece aktarır
- OSI de fiziksel katmanda yer alır.

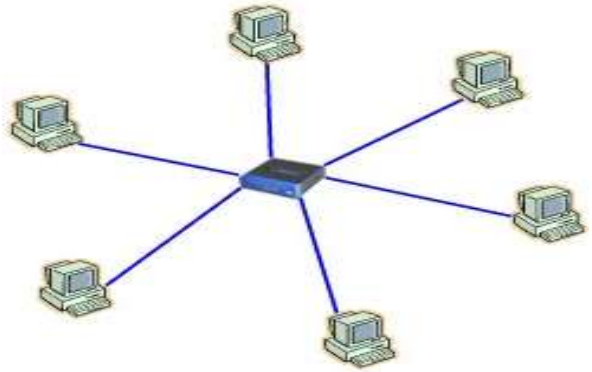
Repeater



The purpose of a repeater is to regenerate and retime network signals at the bit level. This allows them to travel a longer distance on the media.

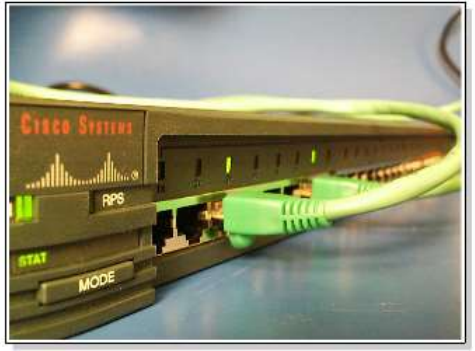
Hub (Dağıtıcı)

- Kablolar ile ağ birimlerinin (bilgisayar vb.) birbirlerine bağlanmasını sağlar.
- Paylaşılan bir yol sunar.
- OSI de Fiziksel katmanda yer alır.
- Port sayısına göre,
- 10/100/1000 Mbps,
- LAN da kullanılır.
- BNC/RJ45.
- Star topoloji.





Switch (Anahtar)

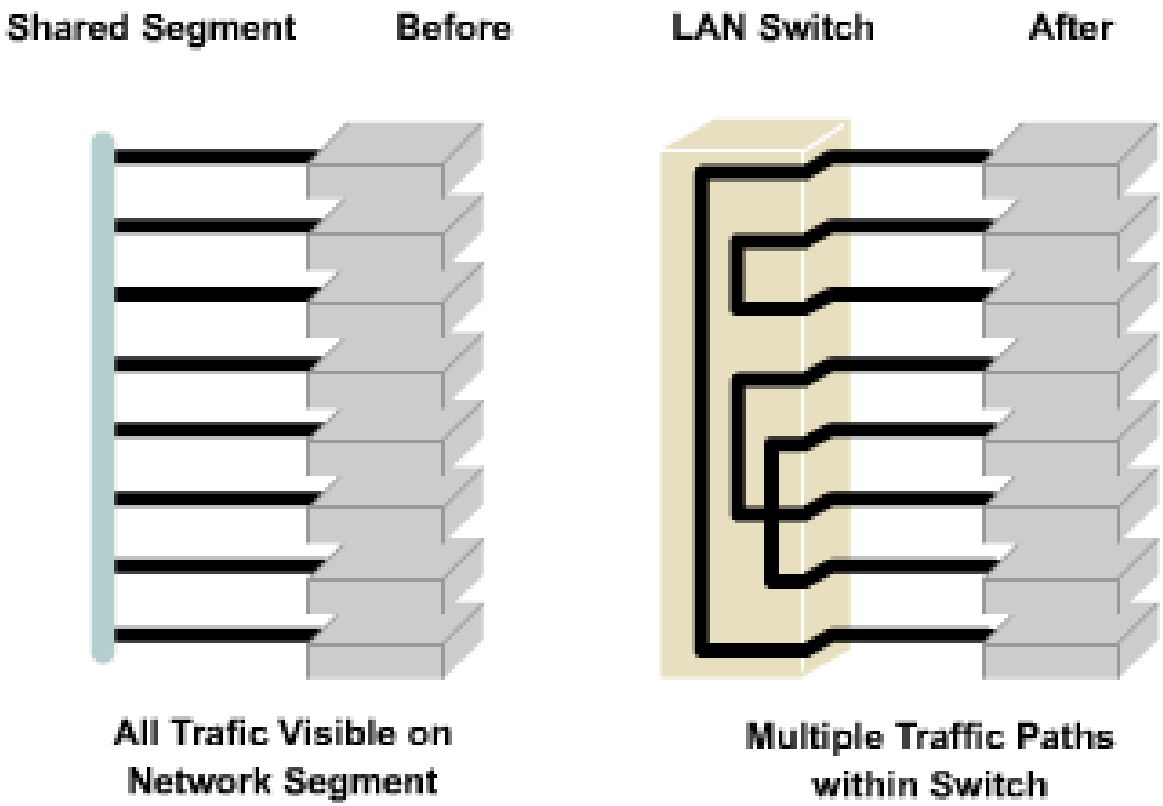


- Kendisine bağlı cihazlara anahtarlamalı bir yol sunar. 8, 12, 16, 24, 36 portlu olabilirler.
- Paket aktarımında MAC adreslerini kullanır.
- Adreslerine göre sadece iki cihazın birbirleri ile haberleşmesine olanak sağlar diğer cihazlar paket trafiğinden etkilenmez.
- Diğer cihazlar kendi aralarında trafiğe devam edebilirler.
- OSI de genelde ikinci katmanda çalışırlar (bazen 3)
- Ethernet, ATM teknolojilerini kullanırlar.

Switch



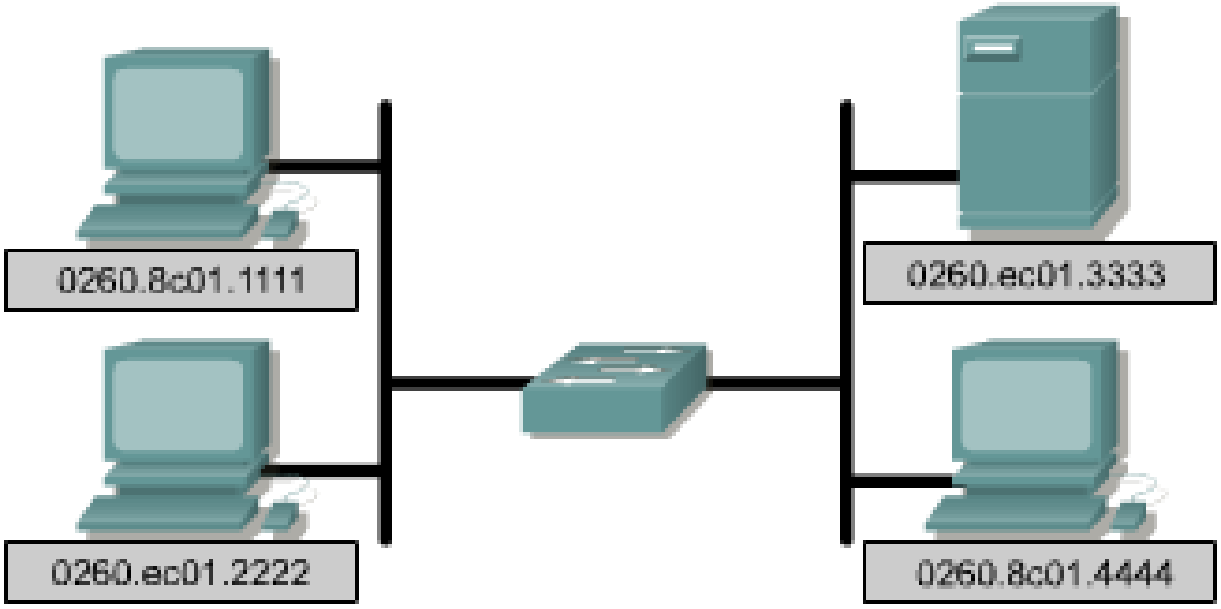
Switch



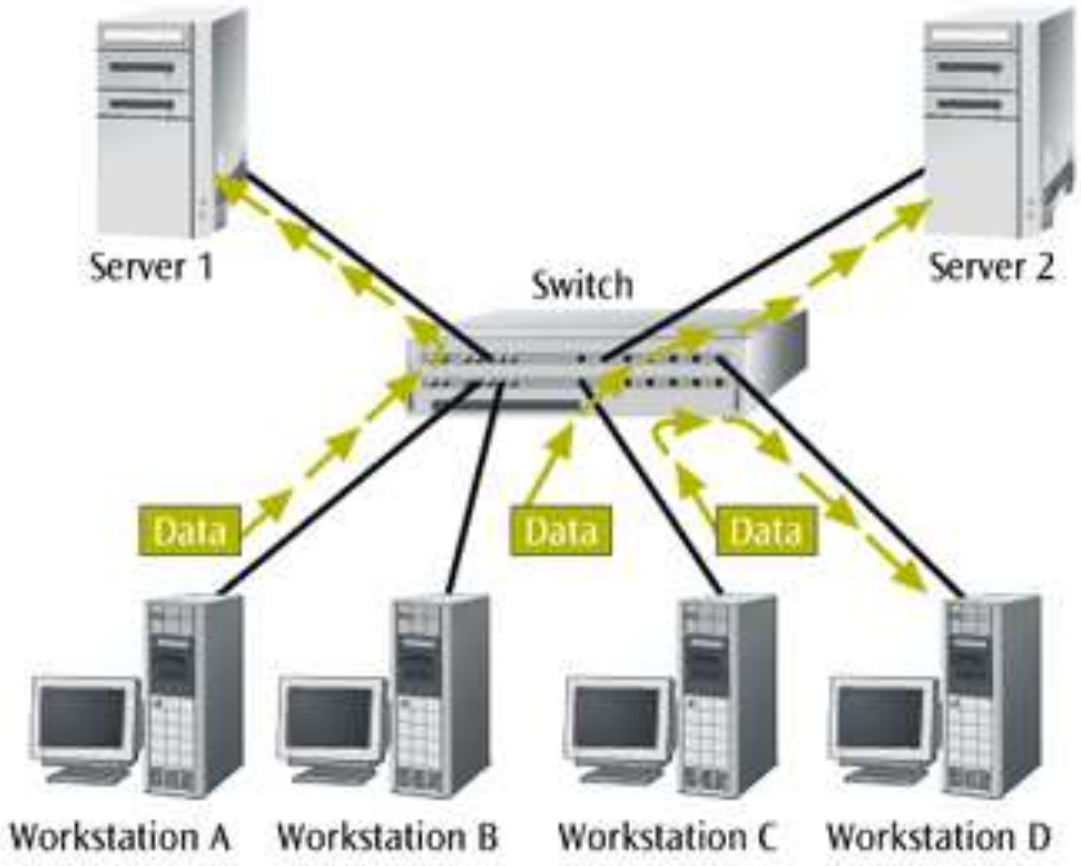
Dedicated paths between sender and receiver hosts.

Switch

Interface	MAC Address
E0	0260.8c01.1111
E0	0260.ec01.2222
E1	0260.ec01.3333
E1	0260.8c01.4444

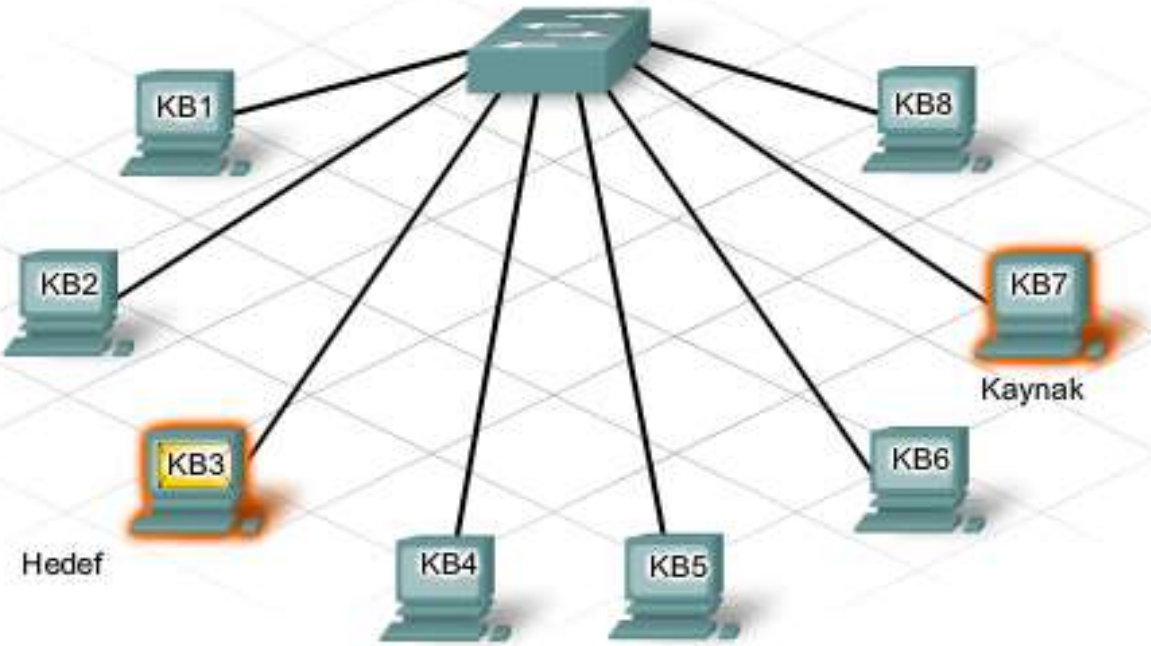


Switch veri akışı



Switch MAC Tablosu

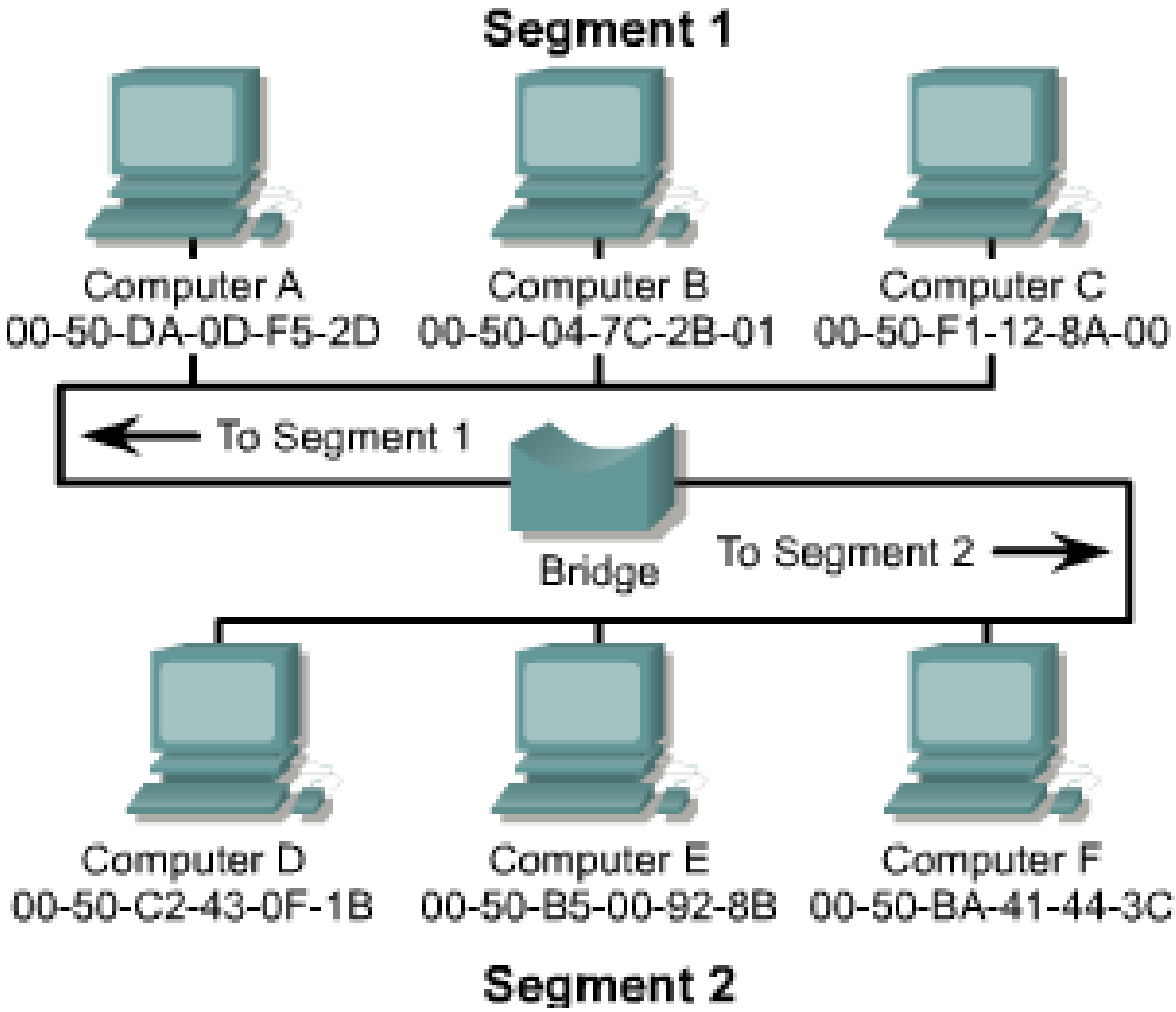
MAC Tablosu			
fa0/1	fa0/2	fa0/3	fa0/4
260d.8c01.0000	260d.8c01.1111	260d.8c01.2222	260d.8c01.3333
fa0/5	fa0/6	fa0/7	fa0/8
260d.8c01.4444	260d.8c01.5555	260d.8c01.6666	260d.8c01.7777



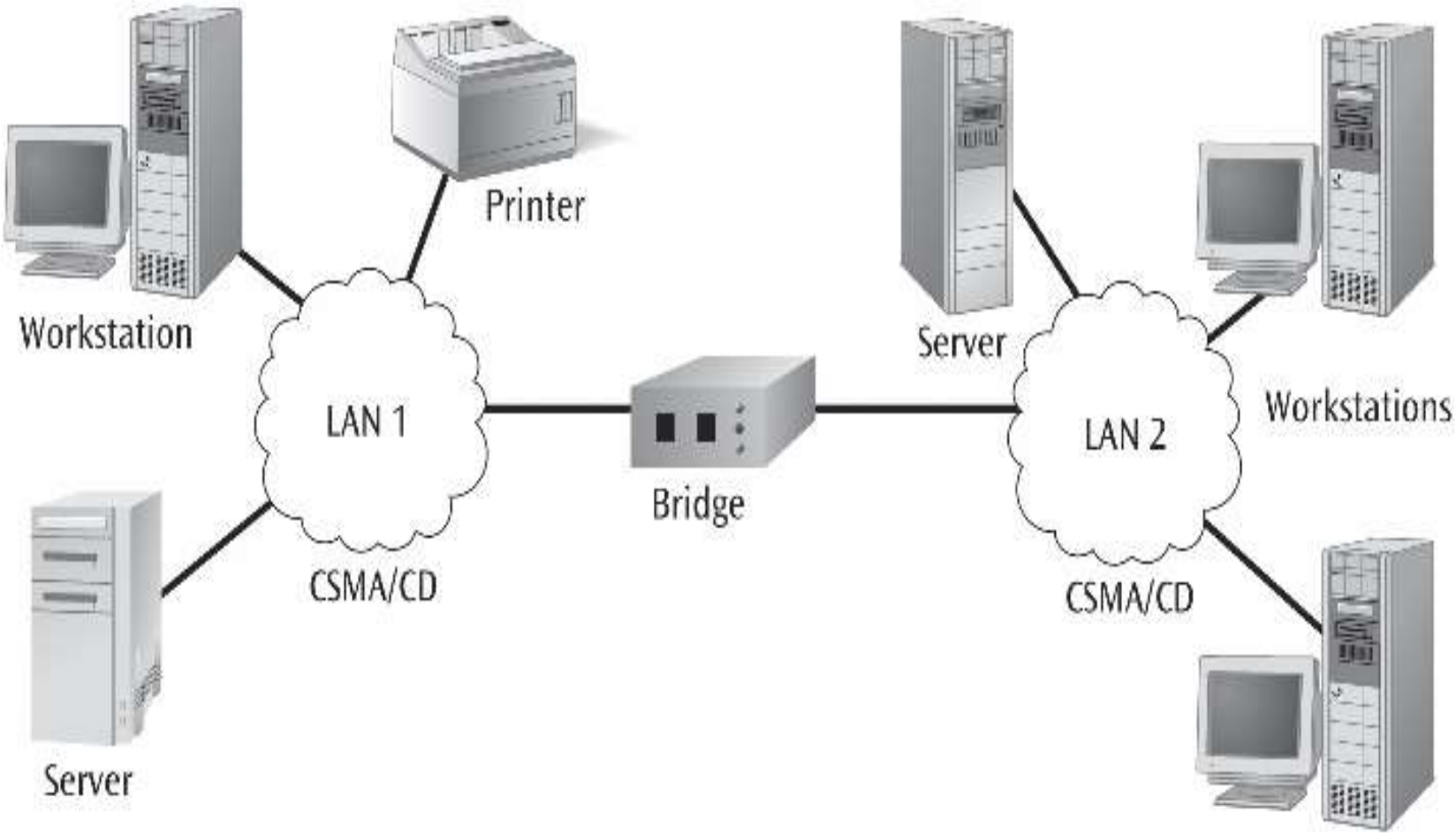
Bridge (Köprü)

- Ağları bölümlere ayırma veya birleştirmede kullanılır. (farklı topolojide olsa)
- MAC adreslerini kullanır.
- OSI de veri iletim katmanında yer alır.
- Trafik yoğunluğunu azaltmayı sağlar.
- Kaynak, Saydam ve çevrimli yöntemleri vardır.
- 10 / 100 Mbps.

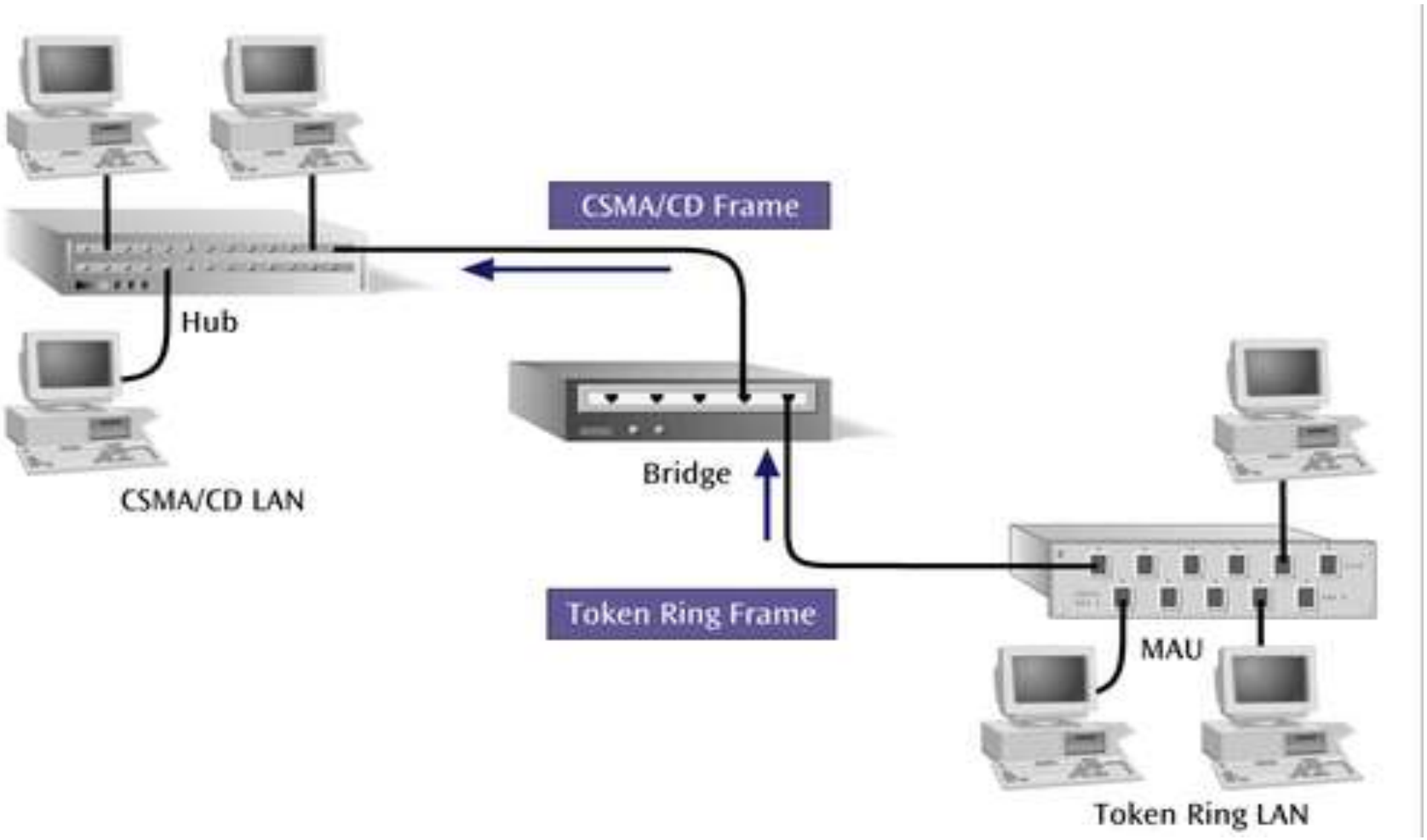
Bridge


Segment 1
Computer A
00-50-DA-0D-F5-2D
Computer B
00-50-04-7C-2B-01
Computer C
00-50-F1-12-8A-00
To Segment 1
Bridge
To Segment 2
Computer D
00-50-C2-43-0F-1B
Computer E
00-50-B5-00-92-8B
Computer F
00-50-BA-41-44-3C
Segment 2

Bridge


Workstation
Printer
LAN 1
CSMA/CD
Server
Bridge
Server
LAN 2
CSMA/CD
Workstations

Bridge


CSMA/CD Frame
Hub
Bridge
CSMA/CD LAN
Token Ring Frame
MAU
Token Ring LAN

Router (Yönlendirici)

- Ağın ve paketlerin yönlendirilmesini sağlar.
- IP adreslerini kullanır. OSI de 3.katmandadır.
- En iyi yolun bulunması işlevini yapar.
- Ağlar arası haberleşme için ara bağlantı sağlar.
- LAN-LAN, LAN-MAN, LAN-WAN da işlev yapar.
- Cihaz, işlemci, ram ve işletim sistemine sahiptir.
- Yönlendirme tablosuna sahiptir.
- Konfigürasyonu yapılabilir. Kurallar vb..
- Farklı portlara (şaseler) sahiptir.
- Statik, Dinamik yönlendirme.





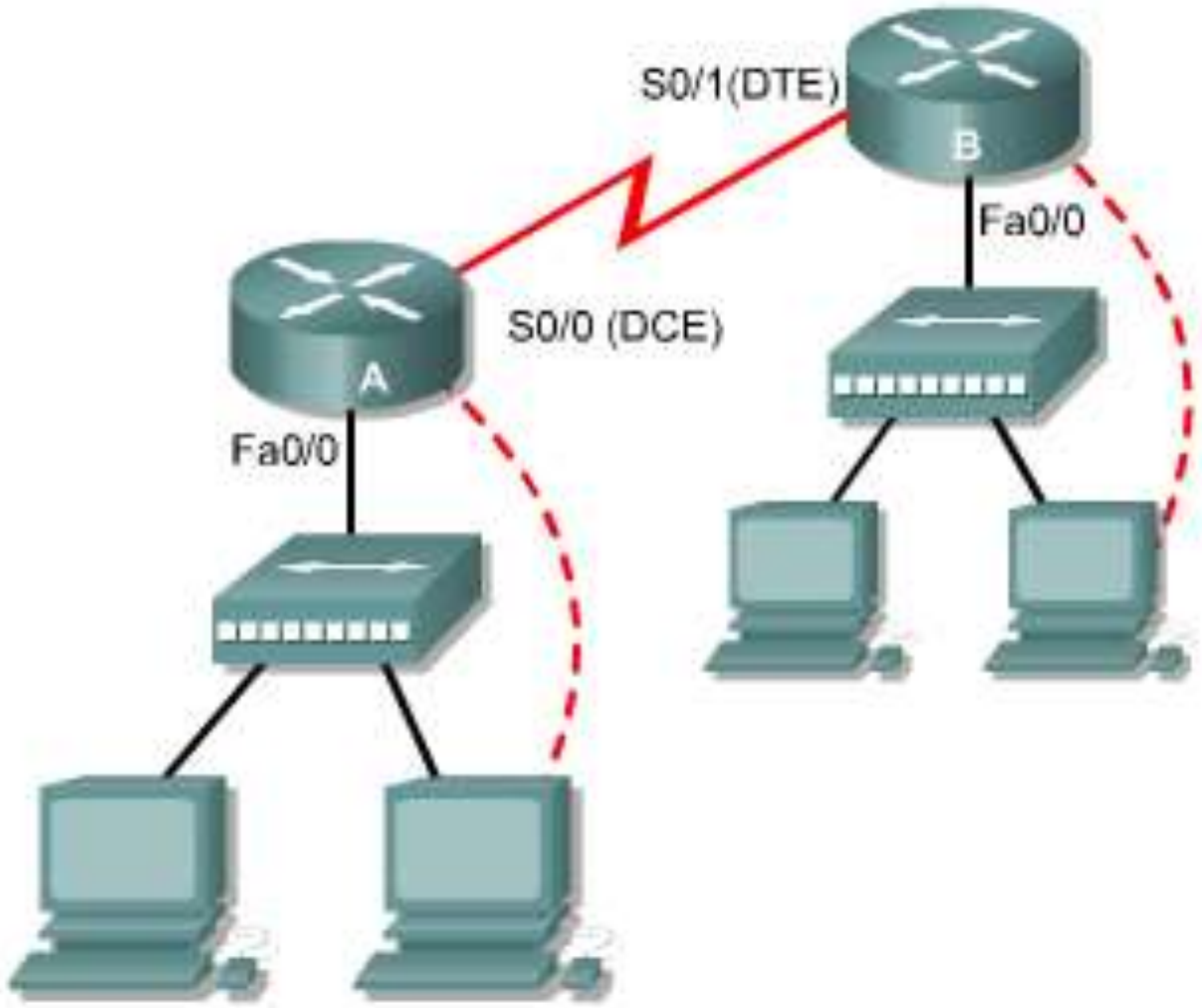




Router



Router



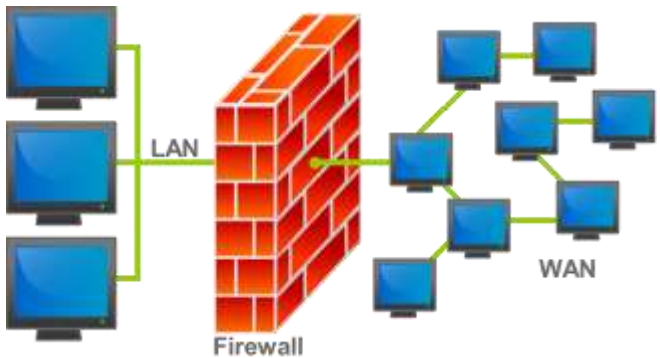
Gateway (Ağ Geçidi)

- Geçityolu olarak da adlandırılır.
- Protokol dönüşümü veya haritalama olanağı sağlayan donanım veya yazılımdır.
- Genelde Routerlar üzerinden tanımlanır.
- Farklı protokol kullanan ağlarda iki yönlü protokol dönüşümü yaparak bağlantı yapılmasını sağlar.
- OSI de tüm katmanları içerir.

Firewall(Ateş Duvarı)



- Ağ erişimine ilişkin içerden veya dışarıdan yetkisiz her erişime engel olmak için veya paketleri süzmek için kullanılan güvenlik amaçlı donanım veya yazılımdır.
- Kurallar tanımlanır. IP ler tanımlanır.
- Portlar kullanılır. Servislere erişim ayarlanır.
- Genelde İzin ver veya yasakla prensibine göre çalışır.

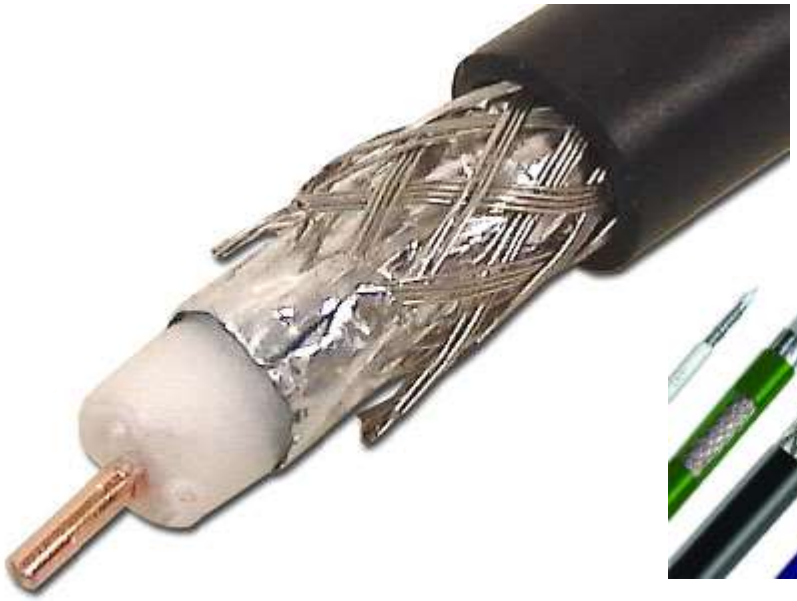




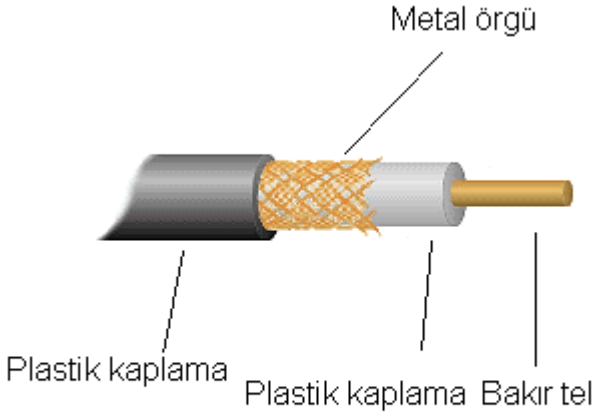
Kablolama Sınıflamaları

- Coaxial (Koaksiyel – Eş eksenli) [BNC]
  - Thin (thinnet) (İnce)
  - Thick (thicknet) (Kalın)
- Twisted-Pair (Çift-bükümlü) [RJ45]
  - STP (Korumalı Çift-bükümlü)
  - UTP (Korumasız Çift-bükümlü)
    - Straight-through (Düz)
    - Cross-over (Ters)
    - Rollover ()
- Fiber-Optik [ST /SC / MT-RJ]

Coaxial(Koaksiyel – Eş eksenli)





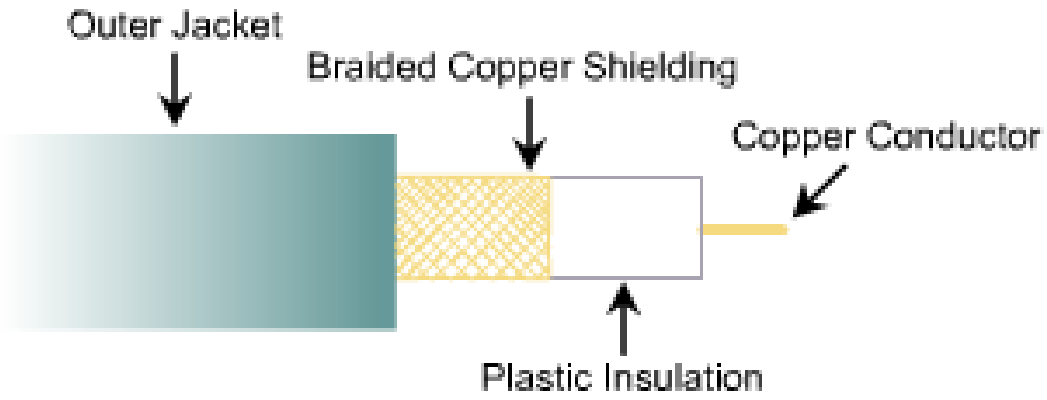


Koaksiyel Kablonun Yapısı

Eşeksenli Kablo

Eşeksenli kablo, ağ kablolaması için geliştirilmiş ilk kablo tiplerinden biridir. Eşeksenli kablo, kablolu TV şirketleri tarafından kullanılan bir bakır kablo tipidir. Eşeksenli kablonun sinyali ileten tek bir sert bakır çekirdeği vardır. Bu çekirdek genellikle kordonlu metal tabaka ve koruyucu kaplamadan oluşan bir izolasyon katmanına sarılır. Yüksek frekans veya geniş bant sinyallerini taşımak için yüksek frekans iletim hattı olarak kullanılır.

Coaxial Cable





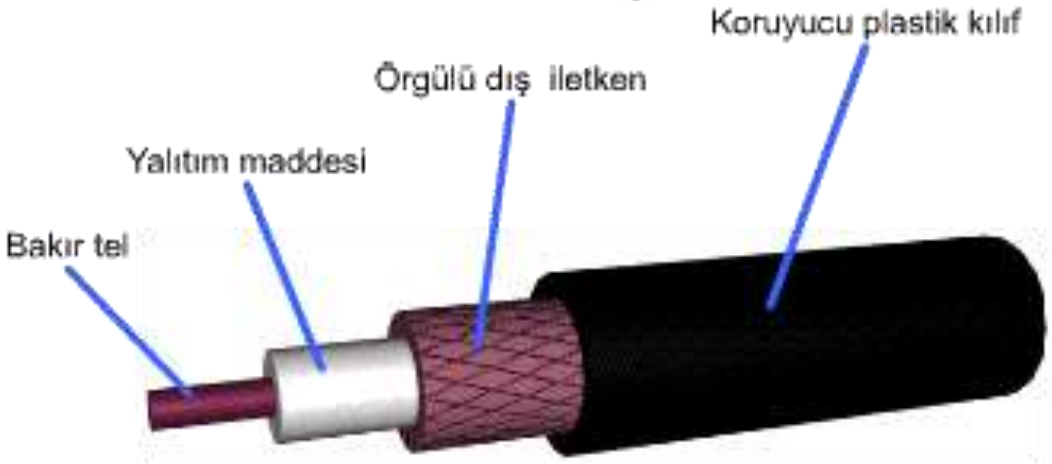
- Speed and throughput: 10 - 100 Mbps
- Average $ per node: Inexpensive
- Media and connector size: Medium
- Maximum cable length: 500m

BNC (British Naval Connector) konnektör

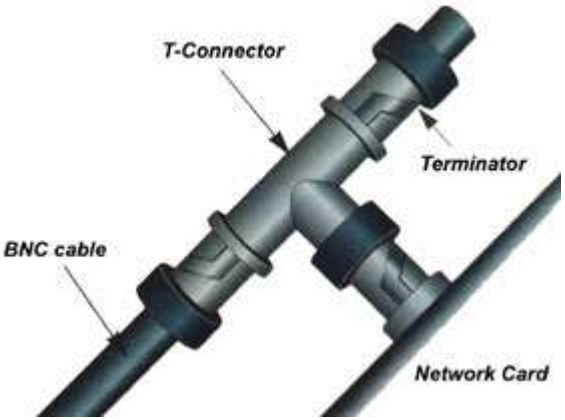




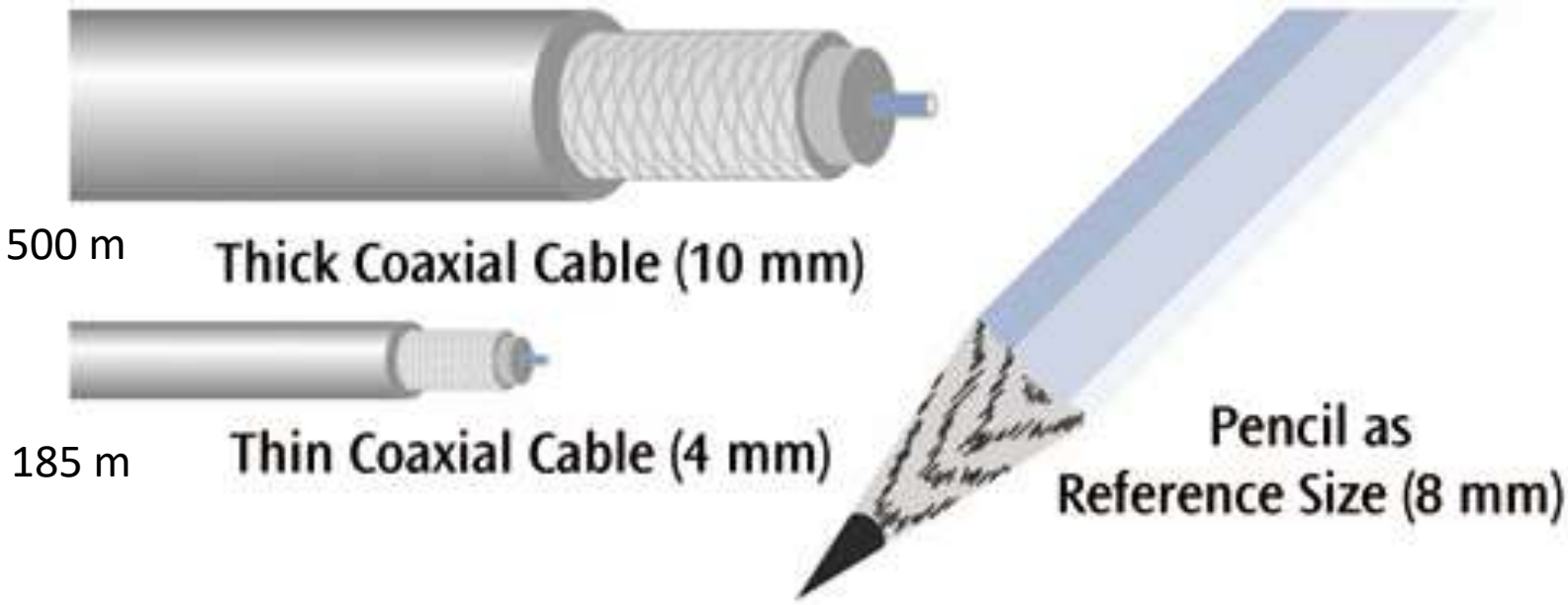




Eşeksenli kablo



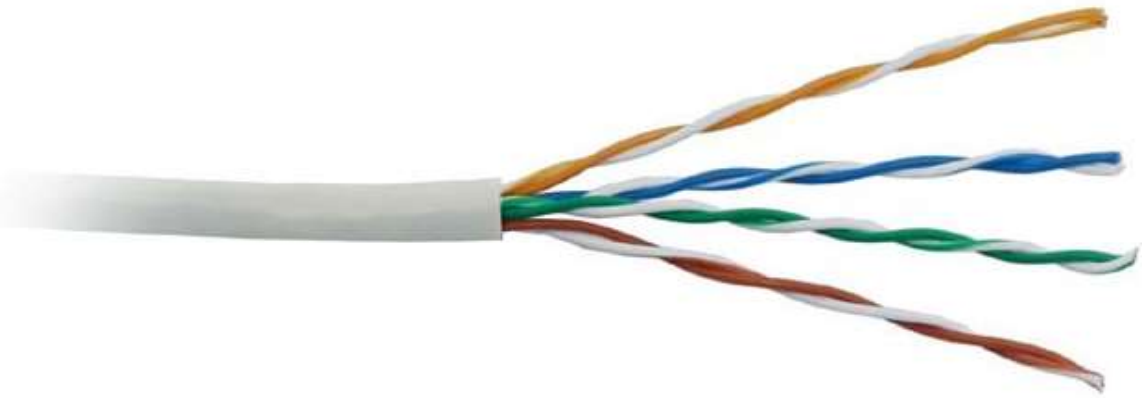
Koaksiyel

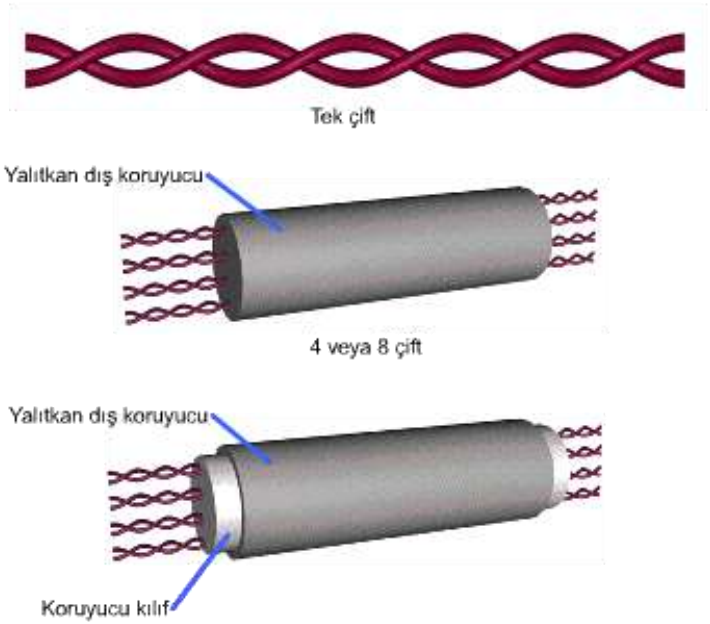

500 m
Thick Coaxial Cable (10 mm)
185 m
Thin Coaxial Cable (4 mm)
Pencil as
Reference Size (8 mm)

Coaxial Cable

TİP	EMPEDANS	KULLANIM
RG-8	**50 Ohm**	10BASE-5 (Kalın-Thicknet) - 500 m
RG-58	**50 Ohm**	10BASE-2* (İnce-Thinnet) - 185 m
RG-59	**75 Ohm**	Kablo TV
RG-6	**75 Ohm**	Anten kablosu

Twisted-Pair (Çift-bükümlü)





Bükümlü Çift

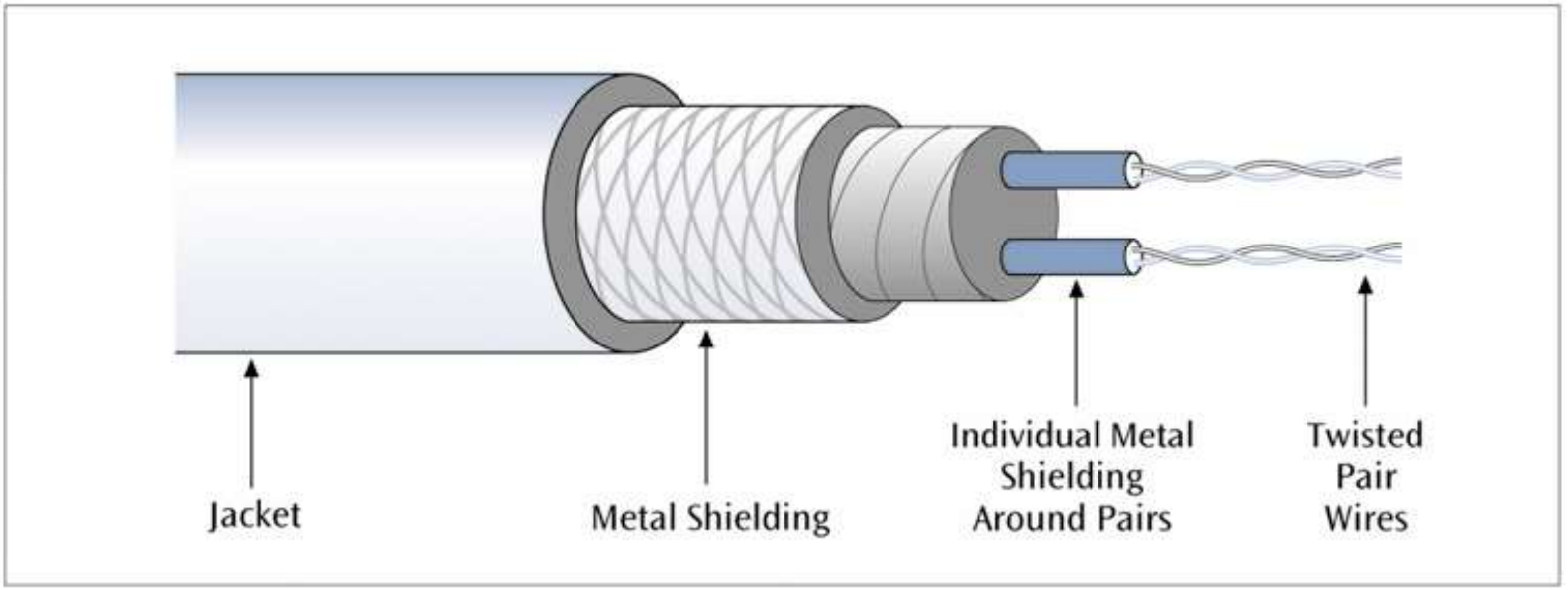
Bükümlü çift (TP) kablo, ağ iletişiminde en sık kullanılan kablo tiplerinden biridir. Girişimi önlemek için teller çiftler halinde gruplanır ve birbirlerine dolanarak bükülür. Her iki uçta da aynı teli ayırt edebilmeniz için tel çiftleri renklendirilir. Genellikle her bir çiftte, tellerden biri düz renk, o telin eşi ise aynı rengin beyaz arkaplan üzerinde çizgili hali olur.

STP - UTP

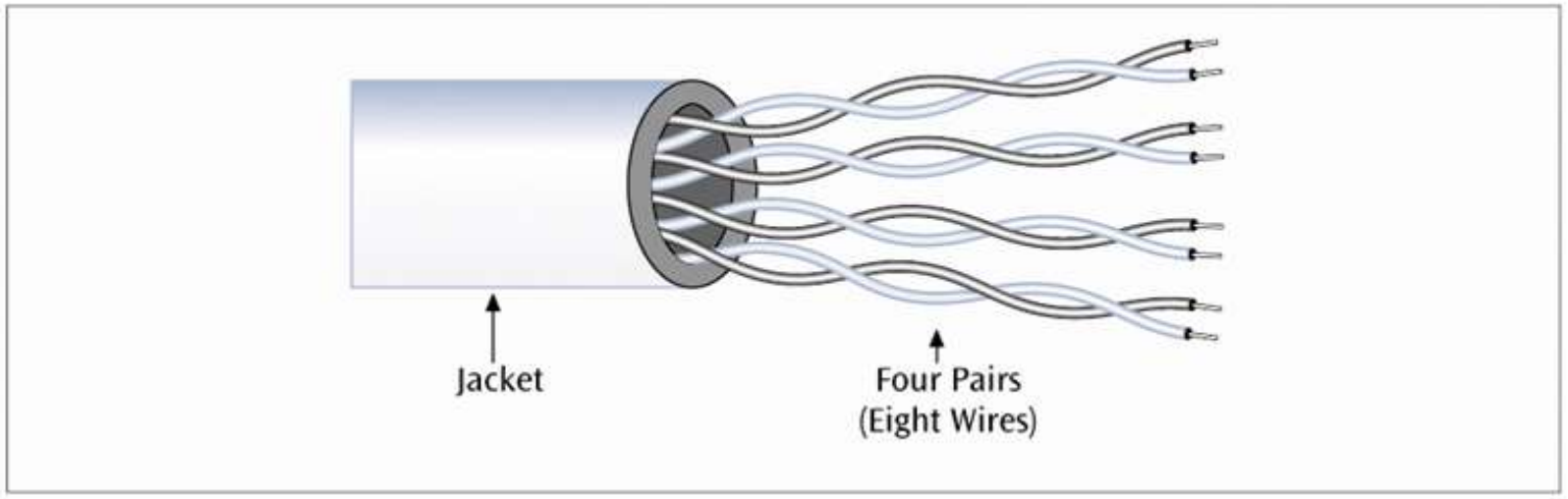
- UTP (Unshielded Twisted Pair)
  Korumasız çift bükümlü

- STP (Shielded Twisted Pair)
  Korumalı çift bükümlü

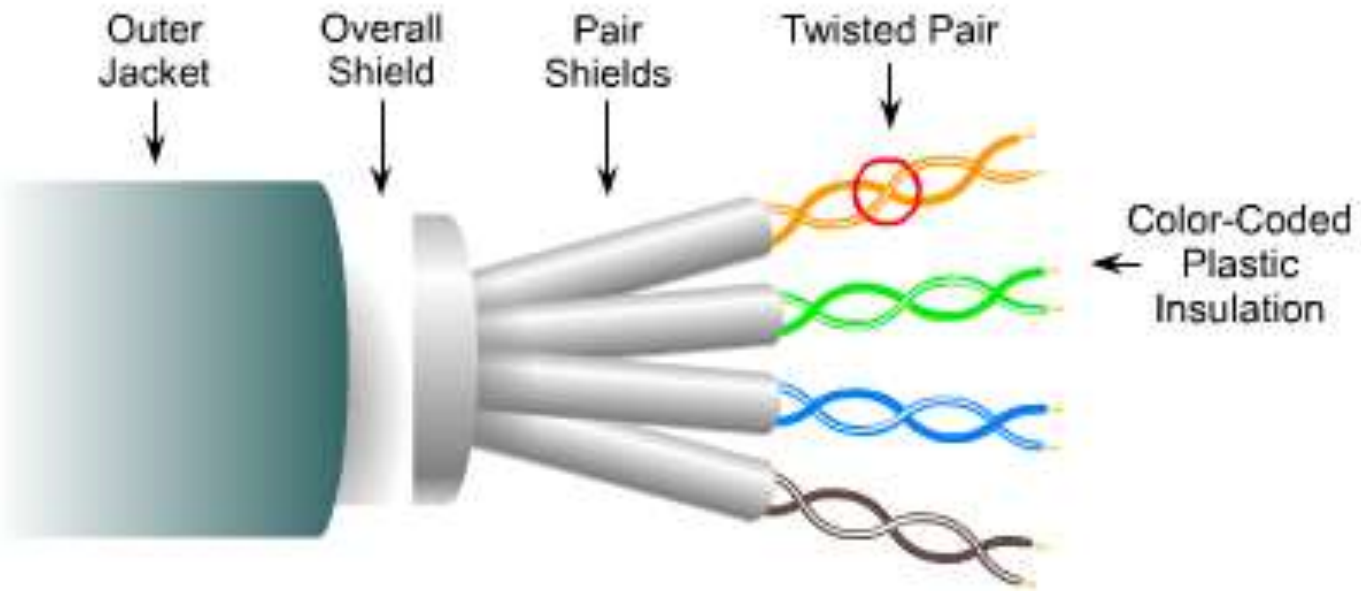
STP


Jacket
Metal Shielding
Individual Metal Shielding Around Pairs
Twisted Pair Wires

UTP

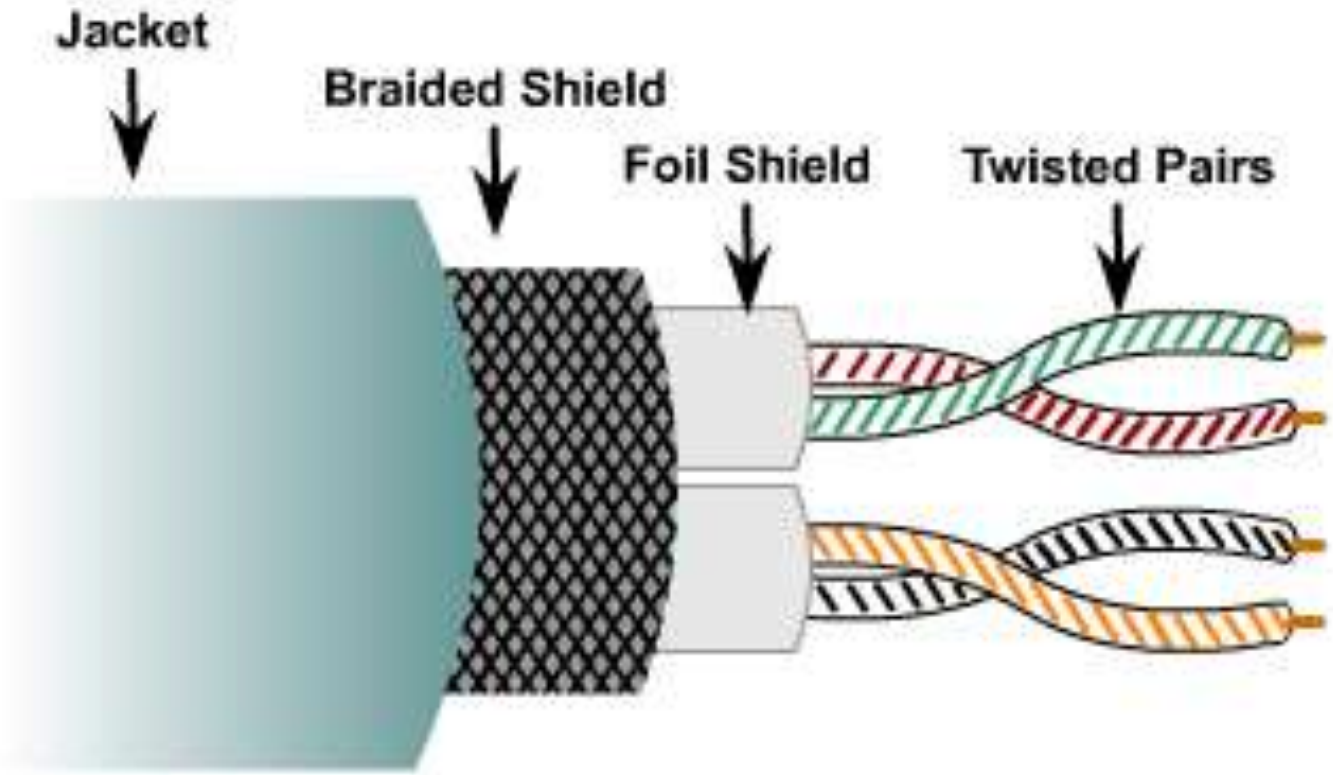

Jacket
Four Pairs (Eight Wires)

Shielded Twisted-Pair Cable

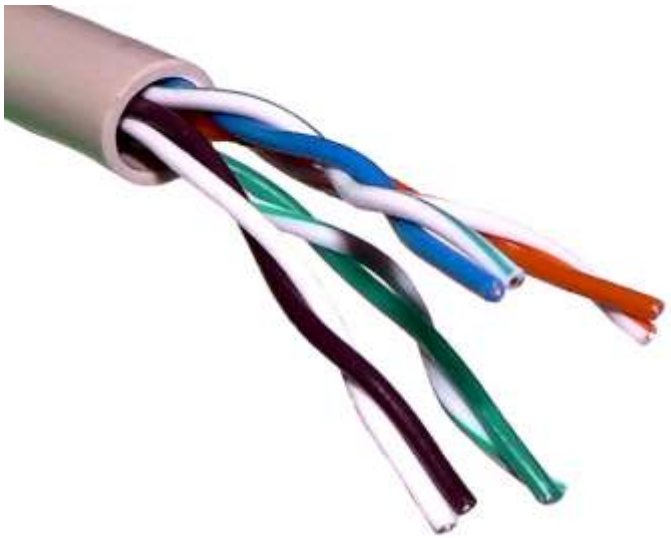


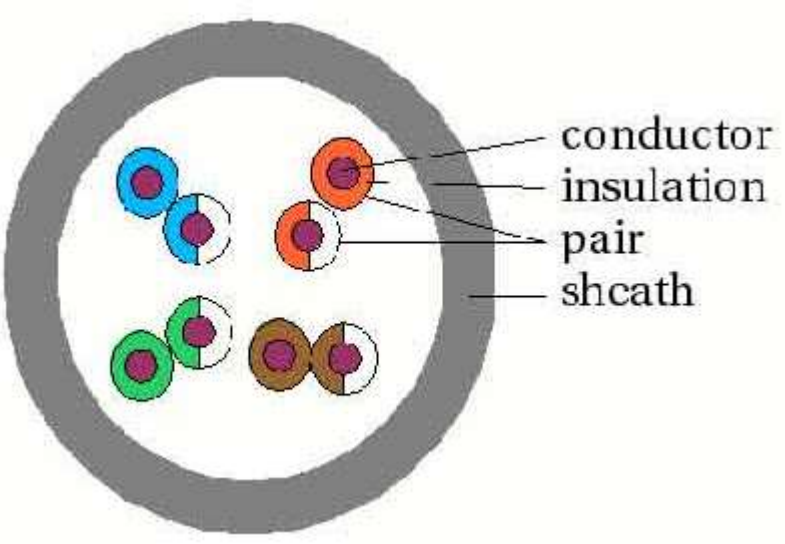
- Speed and throughput: 10 - 100 Mbps
- Average $ per node: Moderately Expensive
- Media and connector size: Medium to Large
- Maximum cable length: 100m

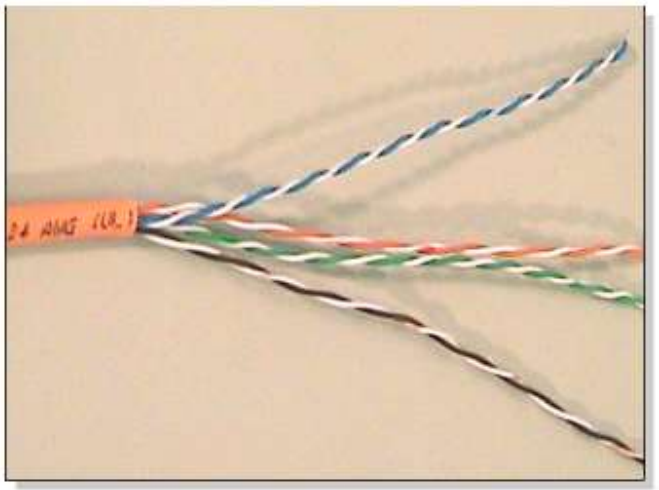
STP



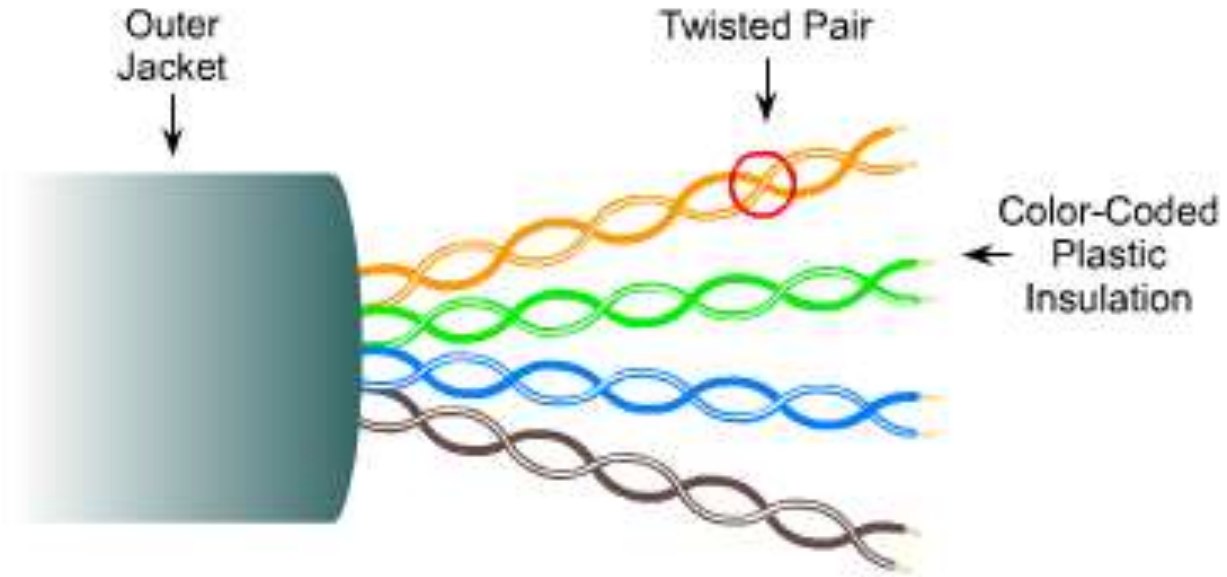
UTP







Unshielded Twisted Pair (UTP)

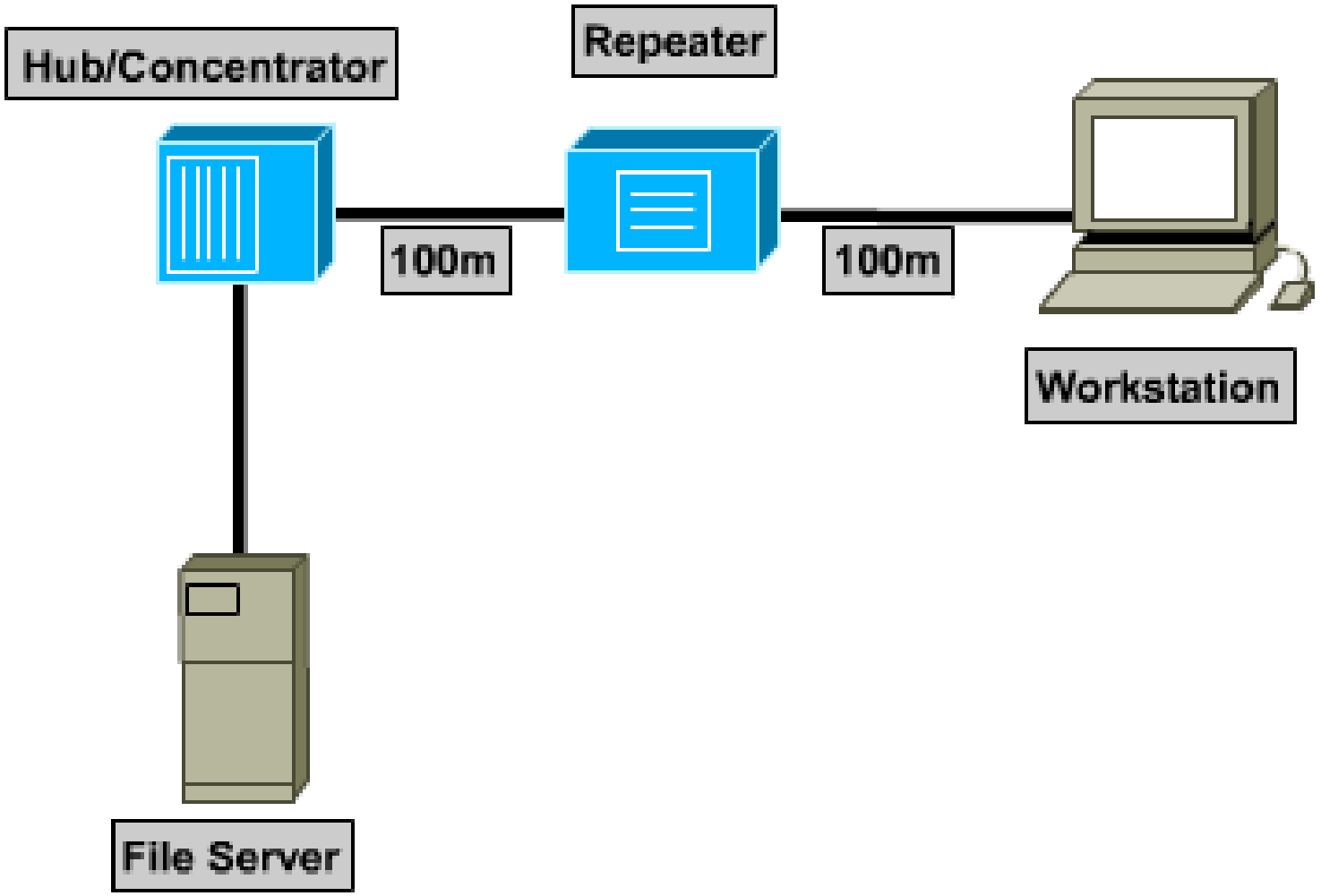


- Speed and throughput: 10 - 100 - 1000 Mbps (depending on the quality/category of cable)
- Average $ per node: Least Expensive
- Media and connector size: Small
- Maximum cable length: 100m

UTP Kablo Kategorileri

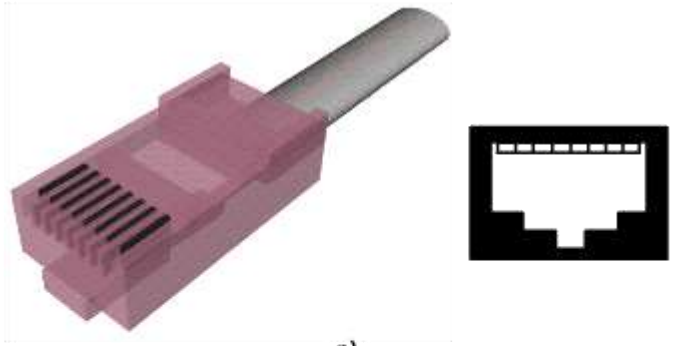
Kategori	Uygulama Alanı
1 (CAT1)	Yanlızca ses veri iletimi yapılmaz
2	Ses ve 1 Mbps' ye kadar veri iletimi.
3	Ses ve 10 Mbps' ye kadar veri iletimi.
4	Ses ve 20 Mbps' ye kadar veri iletimi
5	Ses ve 100 Mbps' ye kadar veri iletimi.
5e	Ses ve 622 Mbps' ye kadar veri iletimi.
6	Ses ve 1 Gps' ye kadar veri iletimi.
7	Ses ve 10 Gps' ye kadar veri iletimi.

UTP


Hub/Concentrator
Repeater
100m
100m
Workstation
File Server

RJ45 Konnektör





a)



b)





RJ45

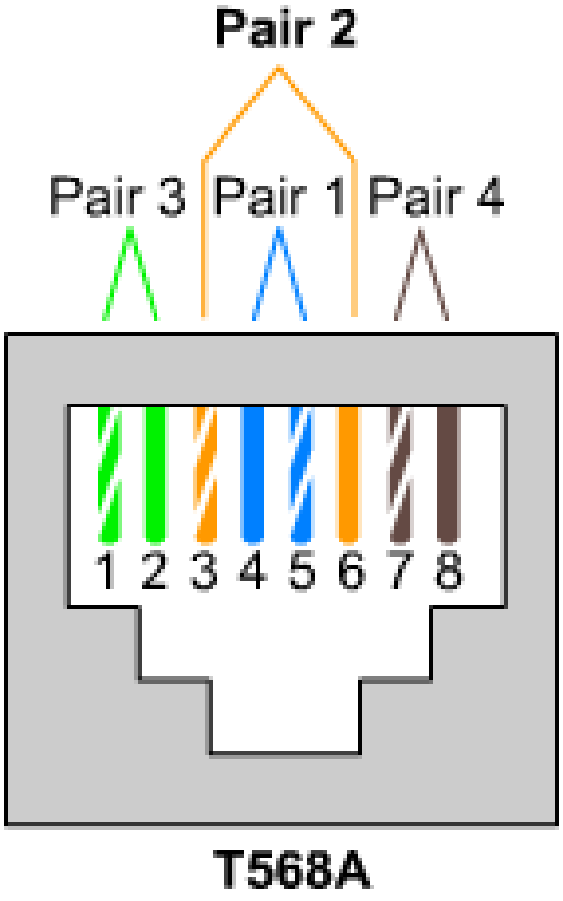


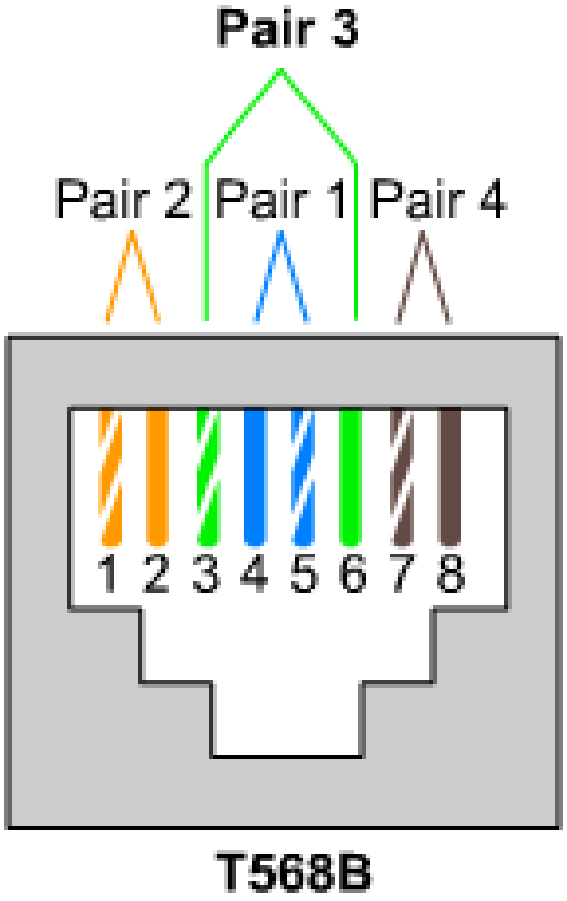
Bad Connector - Wires are untwisted for too great a length.

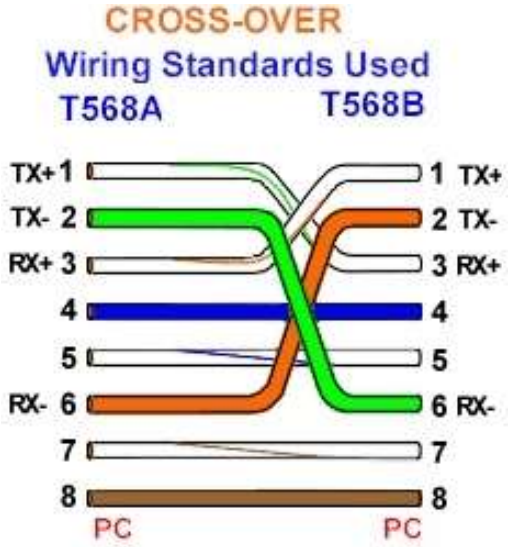


Good Connector - Wires are untwisted to the extent necessary to attach the connector.

T568A – T568B

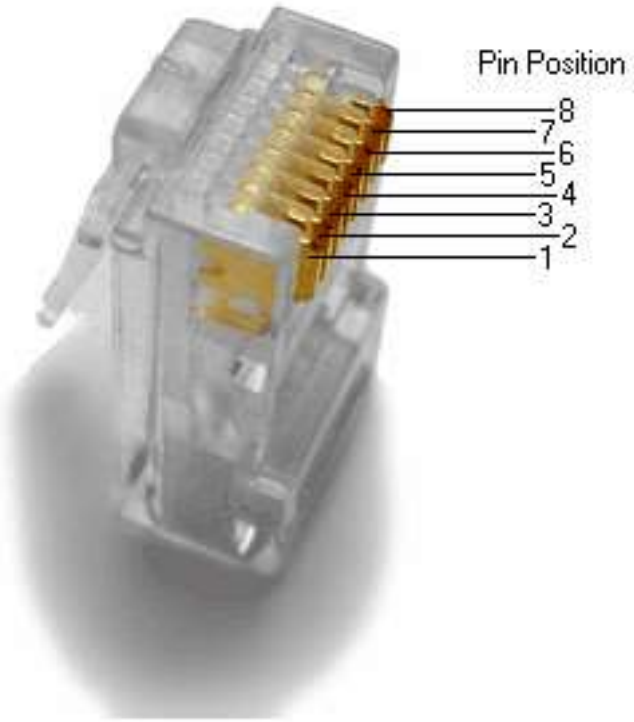

Pair 2
Pair 3
Pair 1
Pair 4
1 2 3 4 5 6 7 8
T568A


Pair 3
Pair 2
Pair 1
Pair 4
1 2 3 4 5 6 7 8
T568B


CROSS-OVER
Wiring Standards Used
T568A
T568B
TX+ 1
TX- 2
RX+ 3
4
5
RX- 6
7
8
1 TX+
2 TX-
3 RX+
4
5
6 RX-
7
8
PC
PC

Pin	T568A Pair	T568B Pair	Wire	T568A Color	T568B Color
1	3	2	tip	white/green stripe	white/orange stripe
2	3	2	ring	green solid	orange solid
3	2	3	tip	white/orange stripe	white/green stripe
4	1	1	ring	blue solid	blue solid
5	1	1	tip	white/blue stripe	white/blue stripe
6	2	3	ring	orange solid	green solid
7	4	4	tip	white/brown stripe	white/brown stripe
8	4	4	ring	brown solid	brown solid

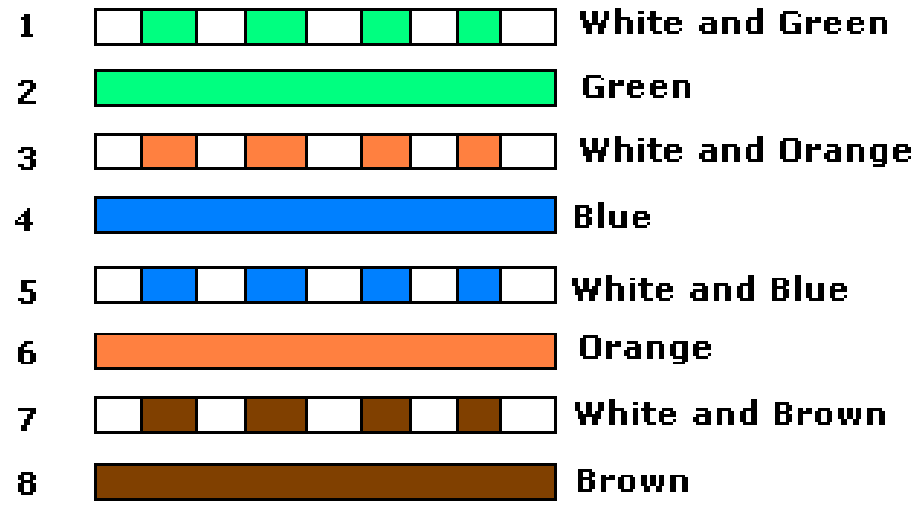
Pins on plug face (socket is reversed)



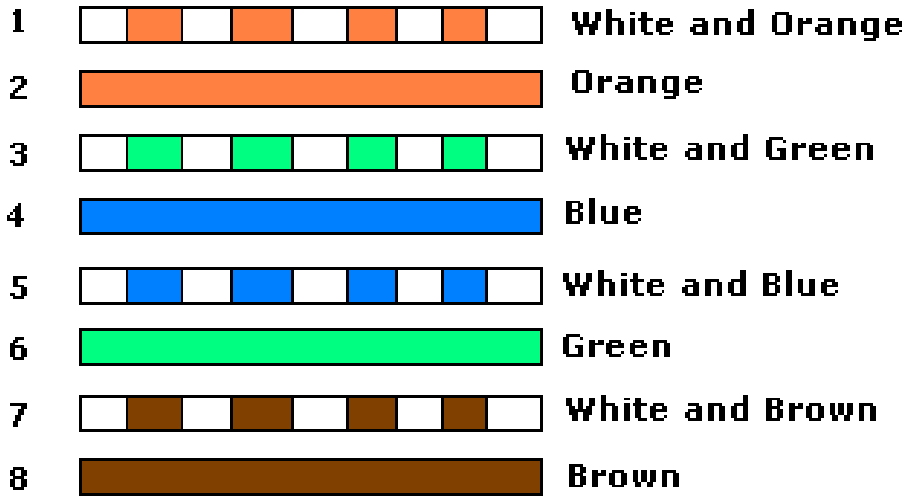
T568A Standardı			
Pin Sayısı	**Tel Çifti Sayısı**	**Kablo Rengi**	**İşlev**
1	2	Beyaz/Yeşil	İletim
2	2	Yeşil	İletim
3	3	Beyaz/Turuncu	Alım
4	1	Mavi	Kullanılmaz
5	1	Beyaz/Mavi	Kullanılmaz
6	3	Turuncu	Alım
7	4	Beyaz/Kahverengi	Kullanılmaz
8	4	Kahverengi	Kullanılmaz

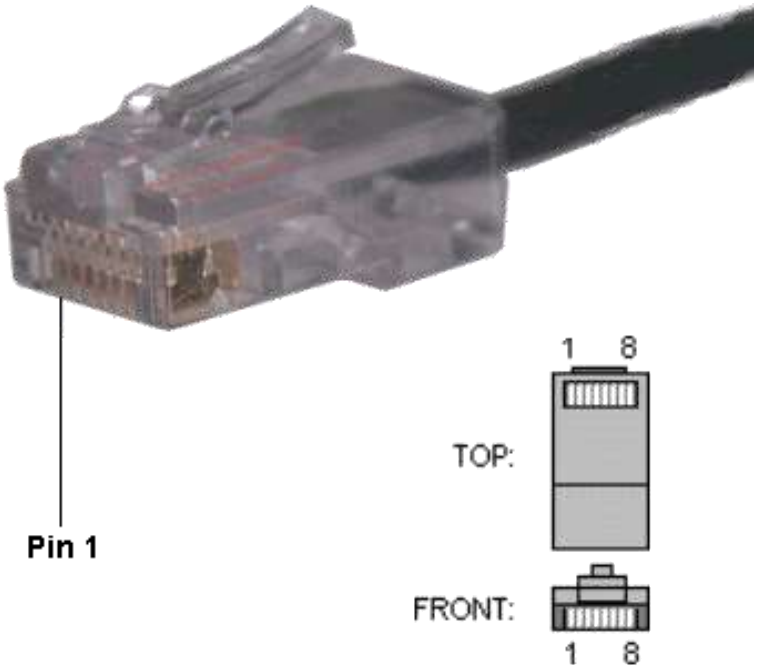
T568B Standardı			
Pin Sayısı	**Tel Çifti Sayısı**	**Kablo Rengi**	**İşlev**
1	2	Beyaz/Turuncu	İletim
2	2	Turuncu	İletim
3	3	Beyaz/Yeşil	Alım
4	1	Mavi	Kullanılmaz
5	1	Beyaz/Mavi	Kullanılmaz
6	3	Yeşil	Alım
7	4	Beyaz/Kahverengi	Kullanılmaz
8	4	Kahverengi	Kullanılmaz

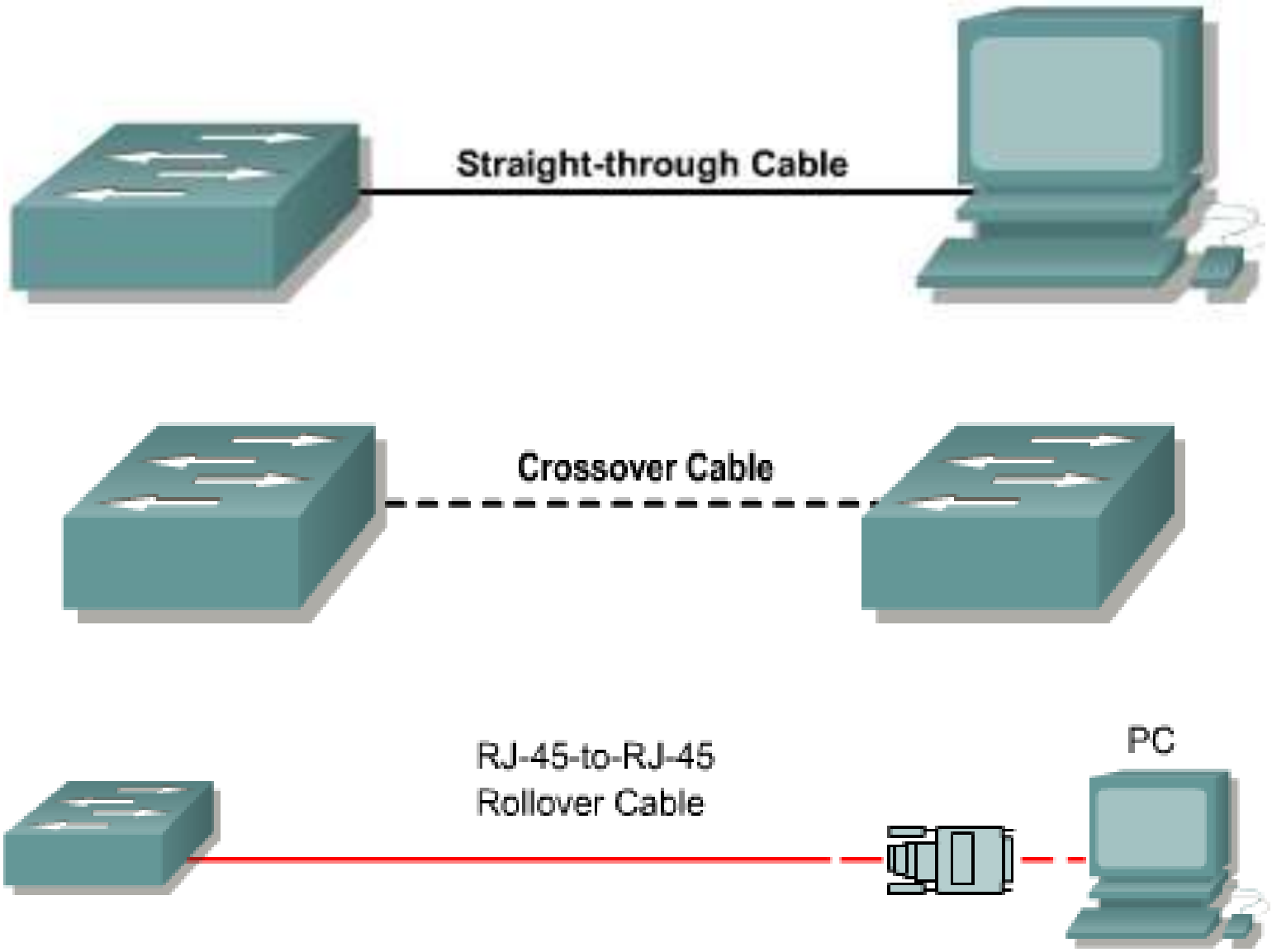
TIA/EIA 568A Wiring



TIA/EIA 568B Wiring

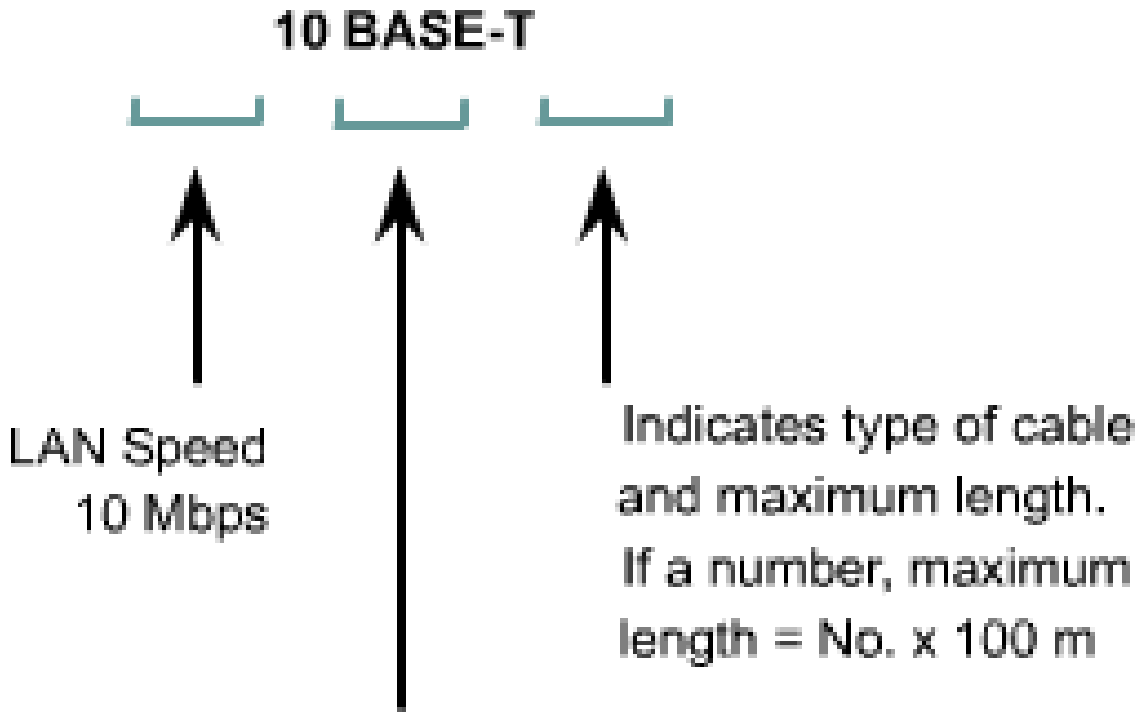





Straight-through Cable
Crossover Cable
RJ-45-to-RJ-45
Rollover Cable
PC
RJ-45-to-DB-9 Adapter
labeled TERMINAL

Ethernet Adı	Kablo Tipi	Max. Veri Transfer Hızı	Max. Veri Transfer Uzaklığı	Açıklama
10Base5	Kalın Koaksiyel	10 Mbps	500 metre	BNC, T
10Base2	İnce Koaksiyel	10 Mbps	185 metre	BNC, T
10BaseT	UTP	10 Mbps	100 metre	RJ-45
100BaseT	UTP	100 Mbps	100 metre	RJ-45
1000BaseT	UTP	1000 Mbps	100 metre	RJ-45, CAT5 ve üstü
1000BaseTX (Gigabit Ethernet)	UTP	1000 Mbps	100 metre	RJ-45, CAT5 ve üstü
10BaseFL	Fiber (multimode)	10 Mbps	2000 metre	Ağlar arası, Fiber optik hub ve NIC arası bağlantı
100BaseFX	Fiber (multimode)	100 Mbps	2000 metre	100 Mbps Ethernet ağlarda
1000BaseSX	Fiber (multimode)	1000 Mbps	260 metre	SC, PC ve hub arası bağlantı için tasarlanmıştır.
1000BaseLX	Fiber (singlemode)	1000 Mbps	550 metre	1000BaseSX'in daha uzun mesafeler arası kullanması için, genellikle omurga olarak kullanılır.

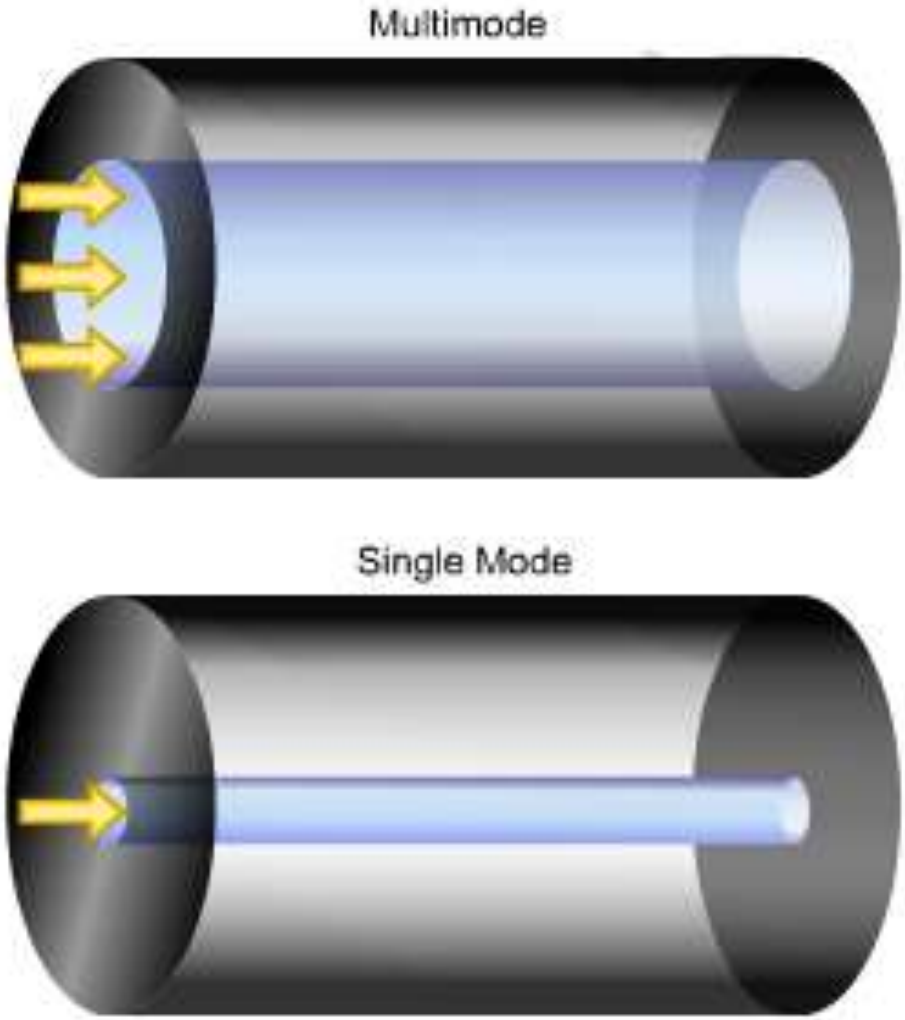
Kablo Özellikleri (İsimlendirme)

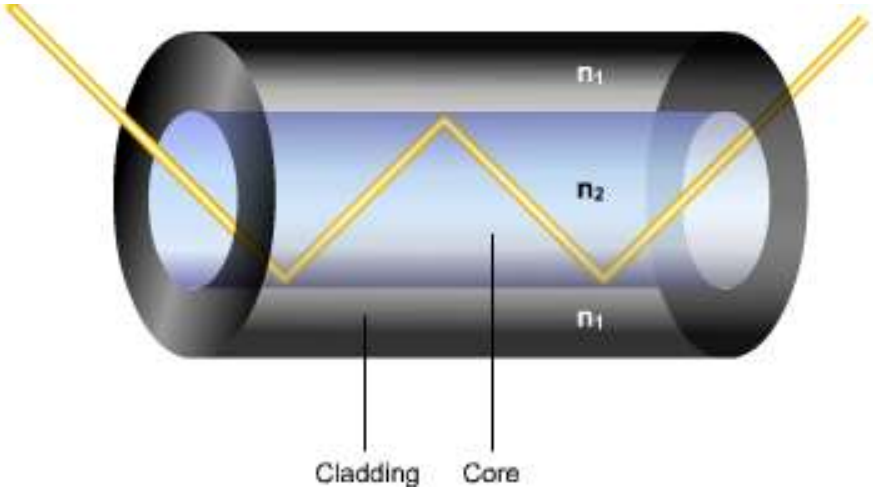


Temel bant – tekli gönderim veya alım

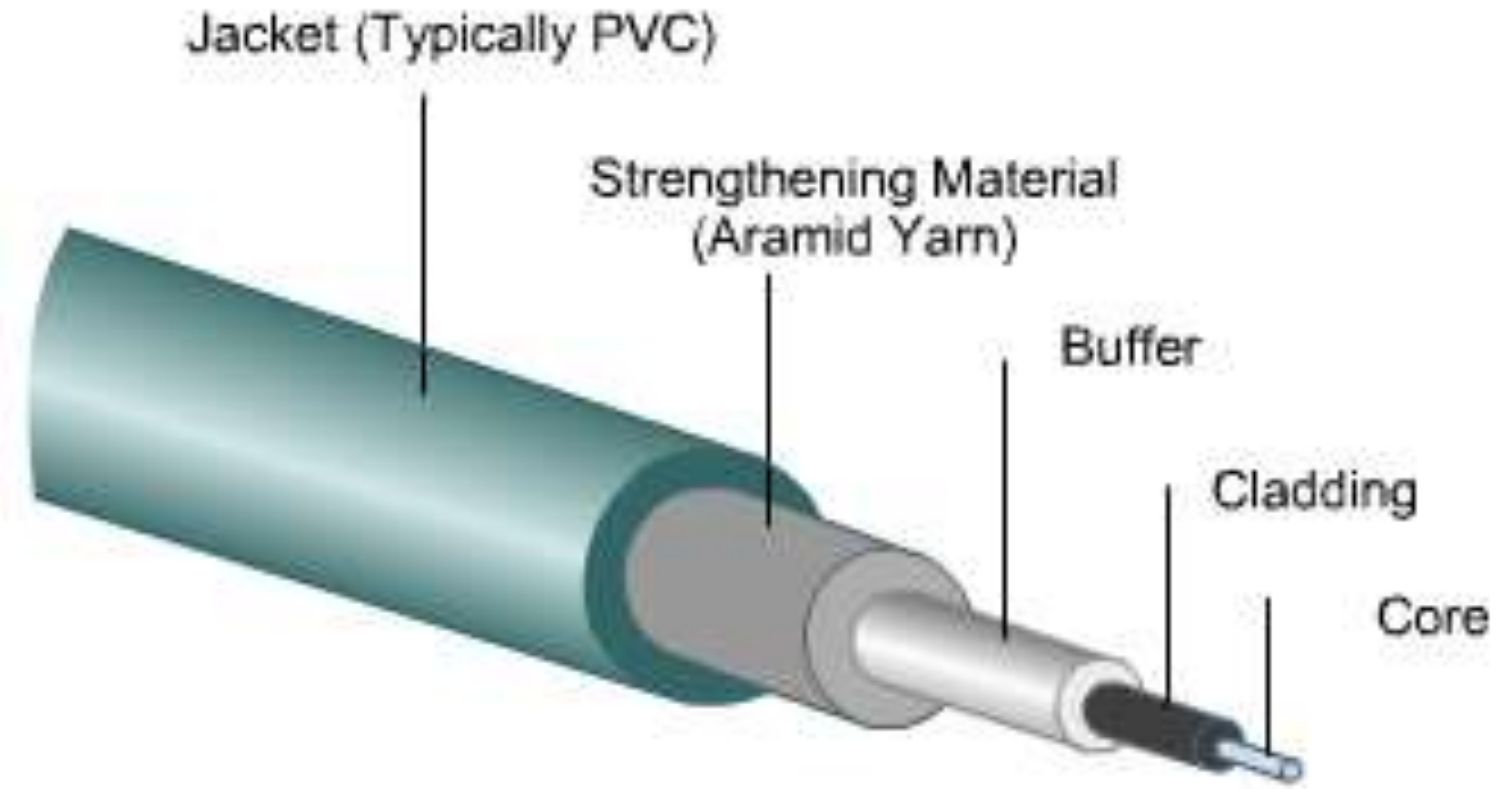
Yayın bant – çoklu gönderim veya alım

Fiber-Optik





Fiber-Optik


Jacket (Typically PVC)
Strengthening Material
(Aramid Yarn)
Buffer
Cladding
Core

Fiber-optik



İç/Dış Ortam Tight Buffer Kablo



İç/Dış Ortam Breakout Kablo



Havai Kablo/Kendinden Destekli



Hibrid ve Kompozit Kablo



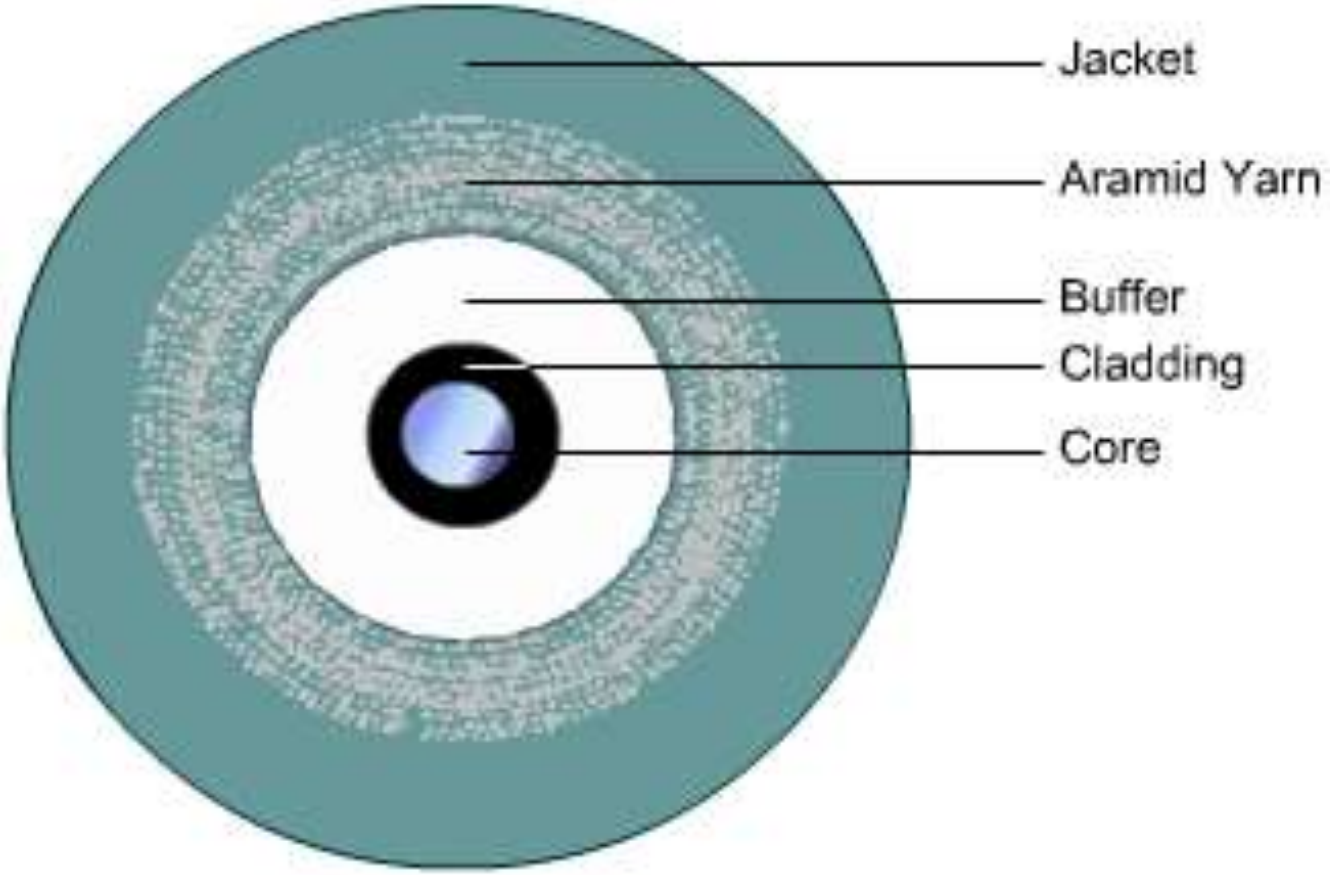
Zırhlı Kablo

Fiber-optik ek cihazı (Füzyon)





Fiber-optik




F-SMA


FDDI/MIC


ESCON


T-ST


T-SC


T-SC-Duplex


T-SC/APC-8°/9°


MT-RJ (male)


MT-RJ (female)


LC


LC-Duplex


FC/PC


FC/APC


DIN


E-2000

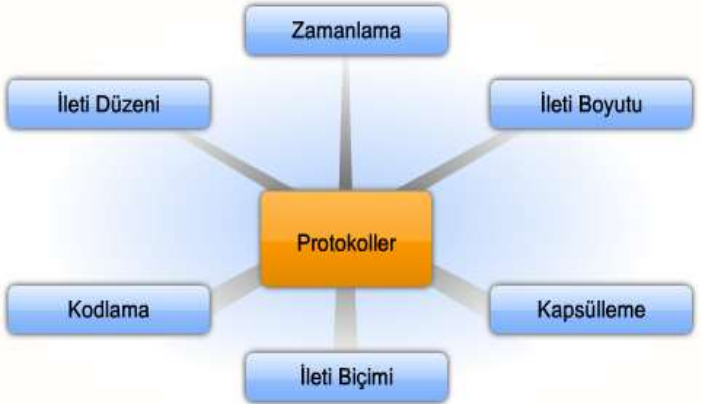

E-2000/APC

6. Hafta

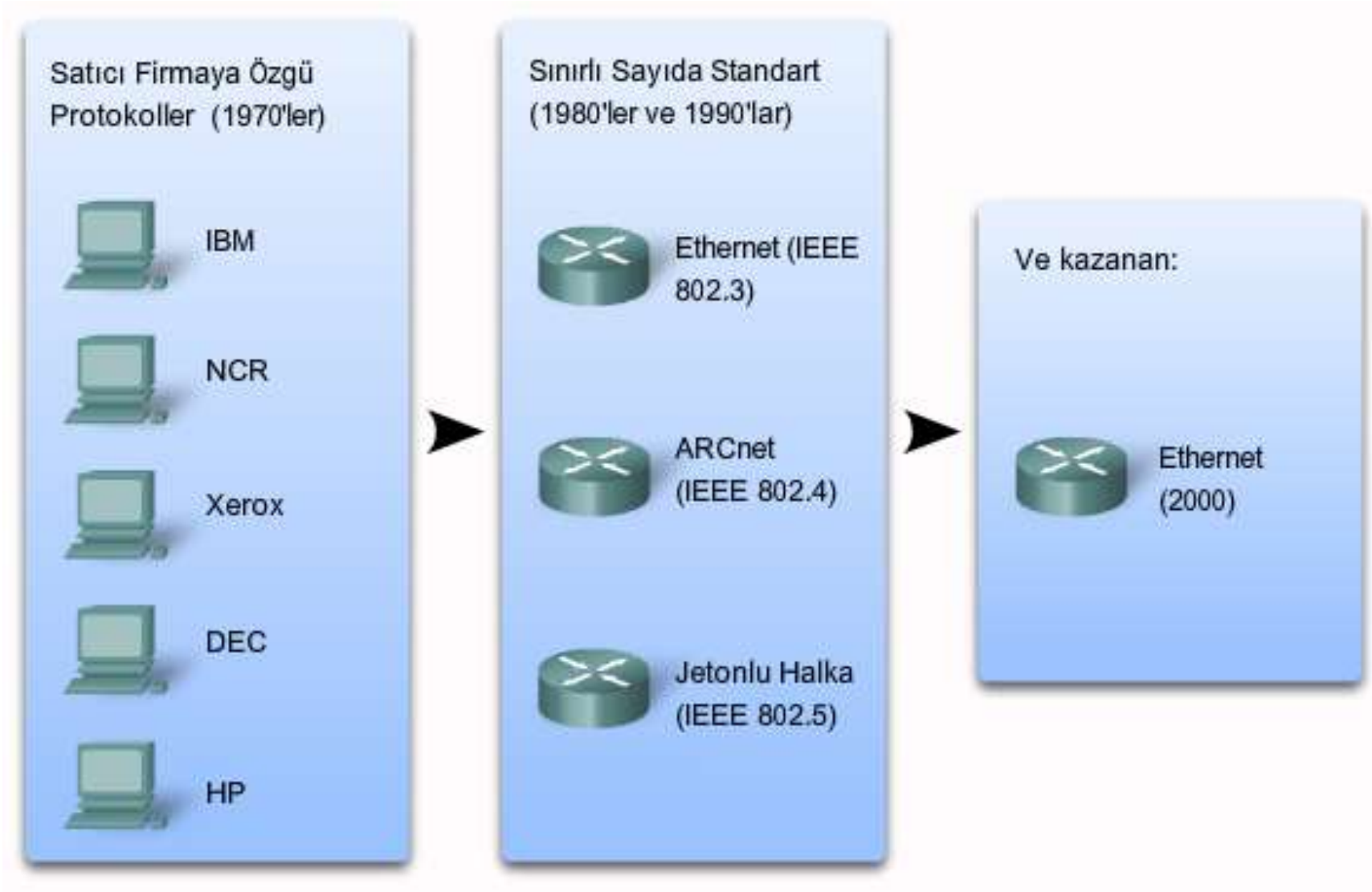
PROTOKOLLER

Protokol

- Protokoller, bilgisayarlar arası iletişimde kullanılan ağ dilleridir. (kurallarıdır.)
- Ağ protokolleri verinin cihazlar arasında nasıl taşınacağını ve ekstra olarak veri ile hangi bilgilerin gönderileceğini belirler.
- En sık kullanılan ve bilinen protokol TCP/IP protokol grubudur.
  - Internet erişimi tamamen TCP/IP'ye dayanır.



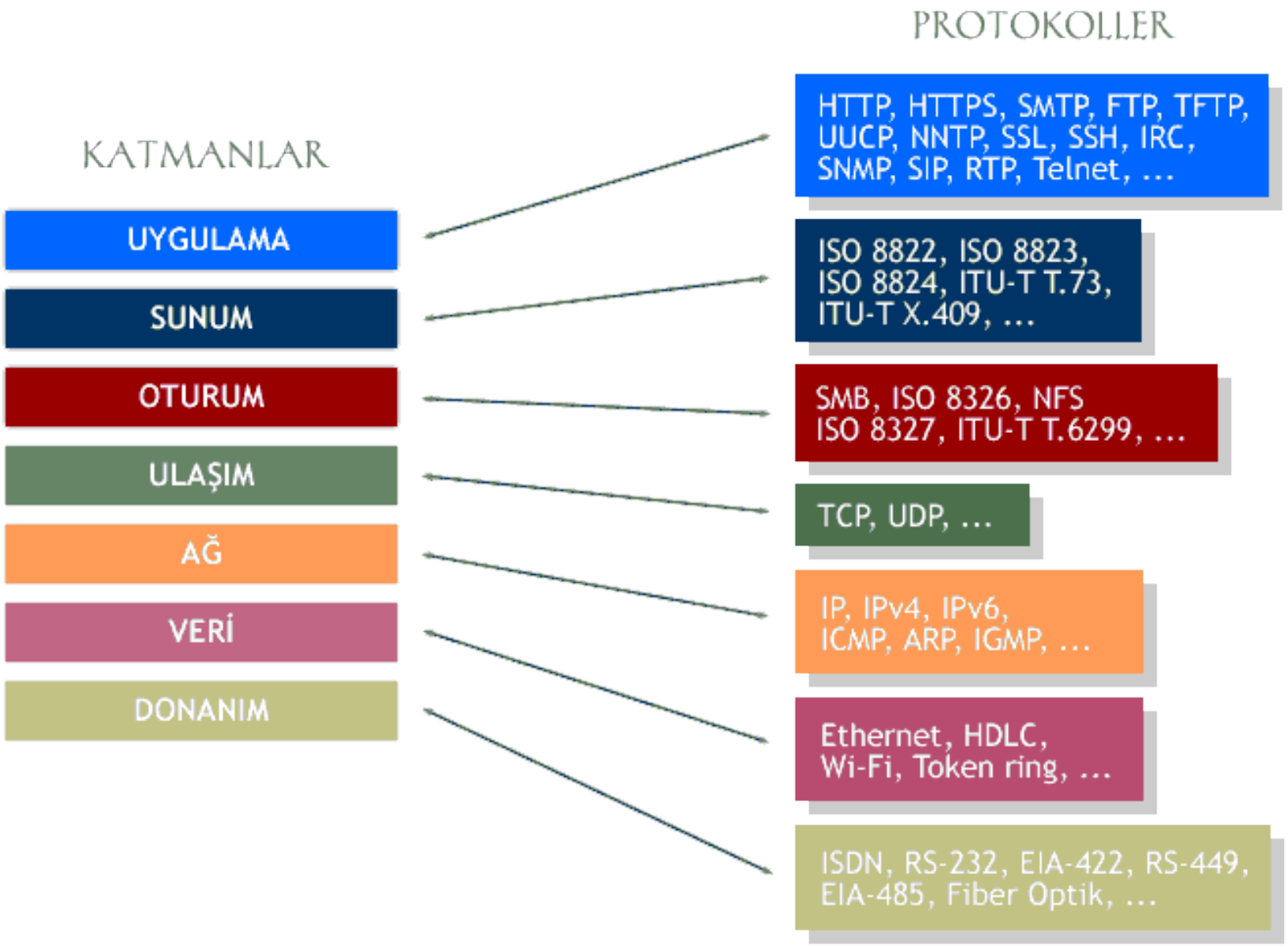
Protokollerin Standartlaştırılması

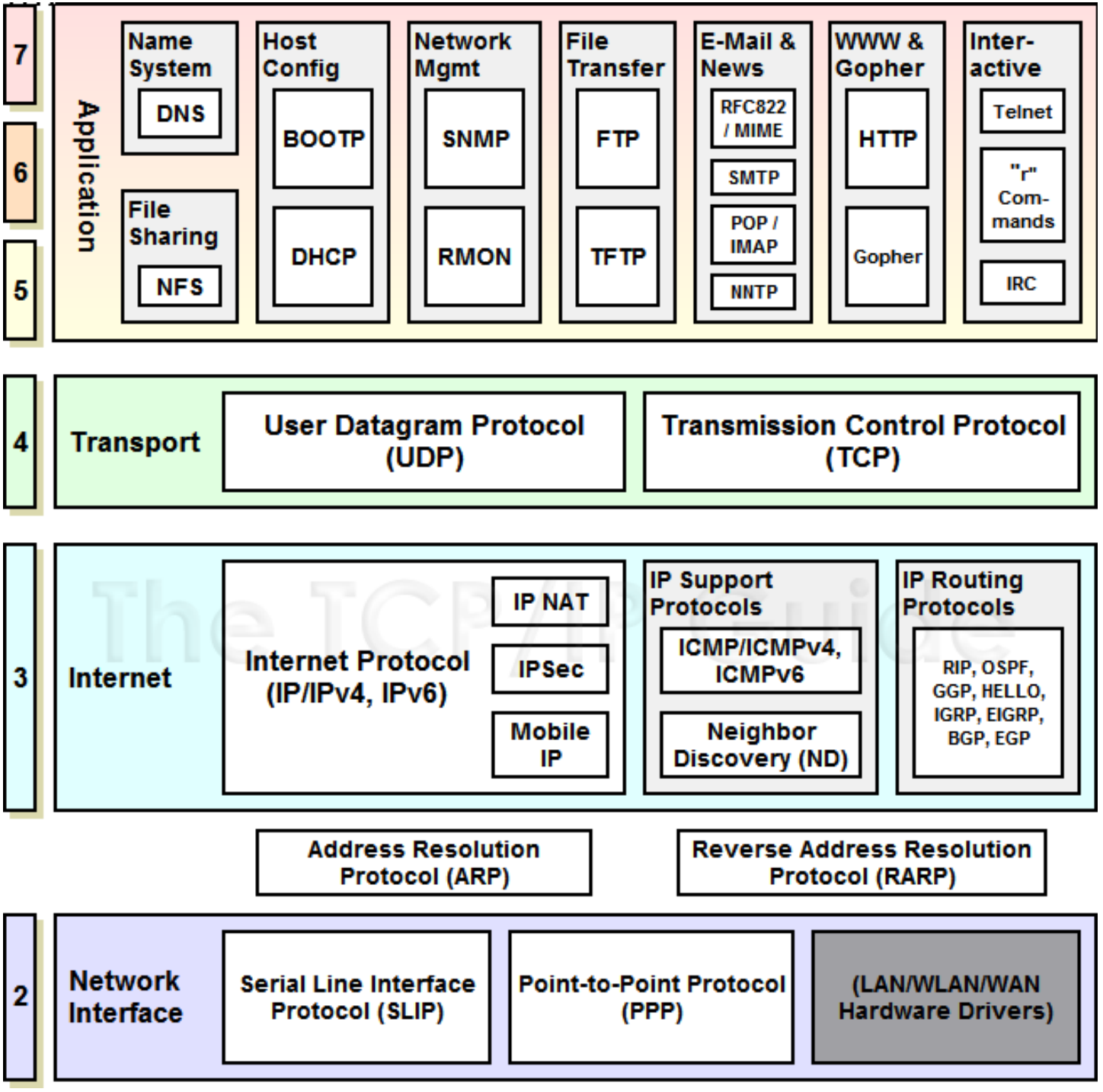


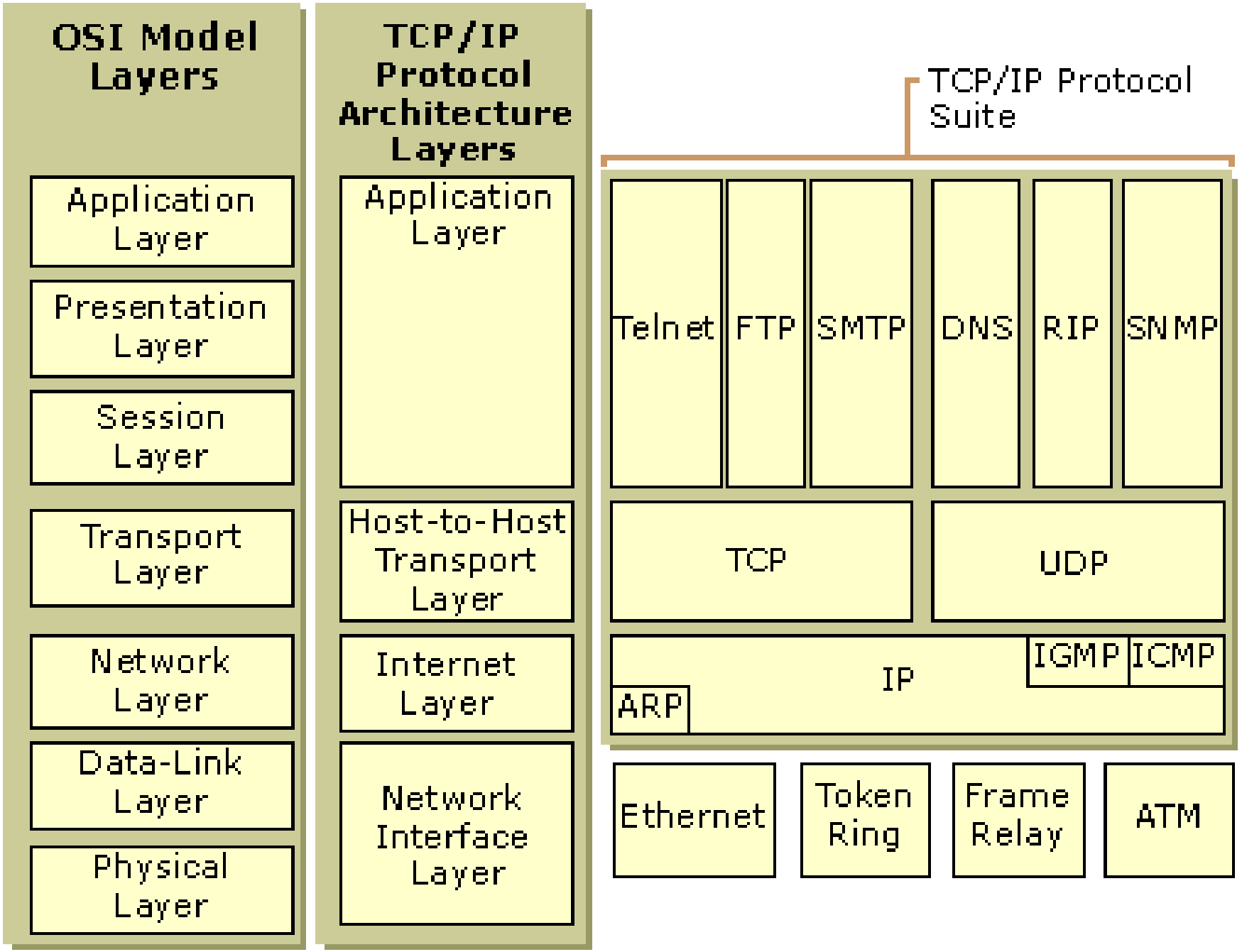
Protokol Kümeleri(Yığınları)

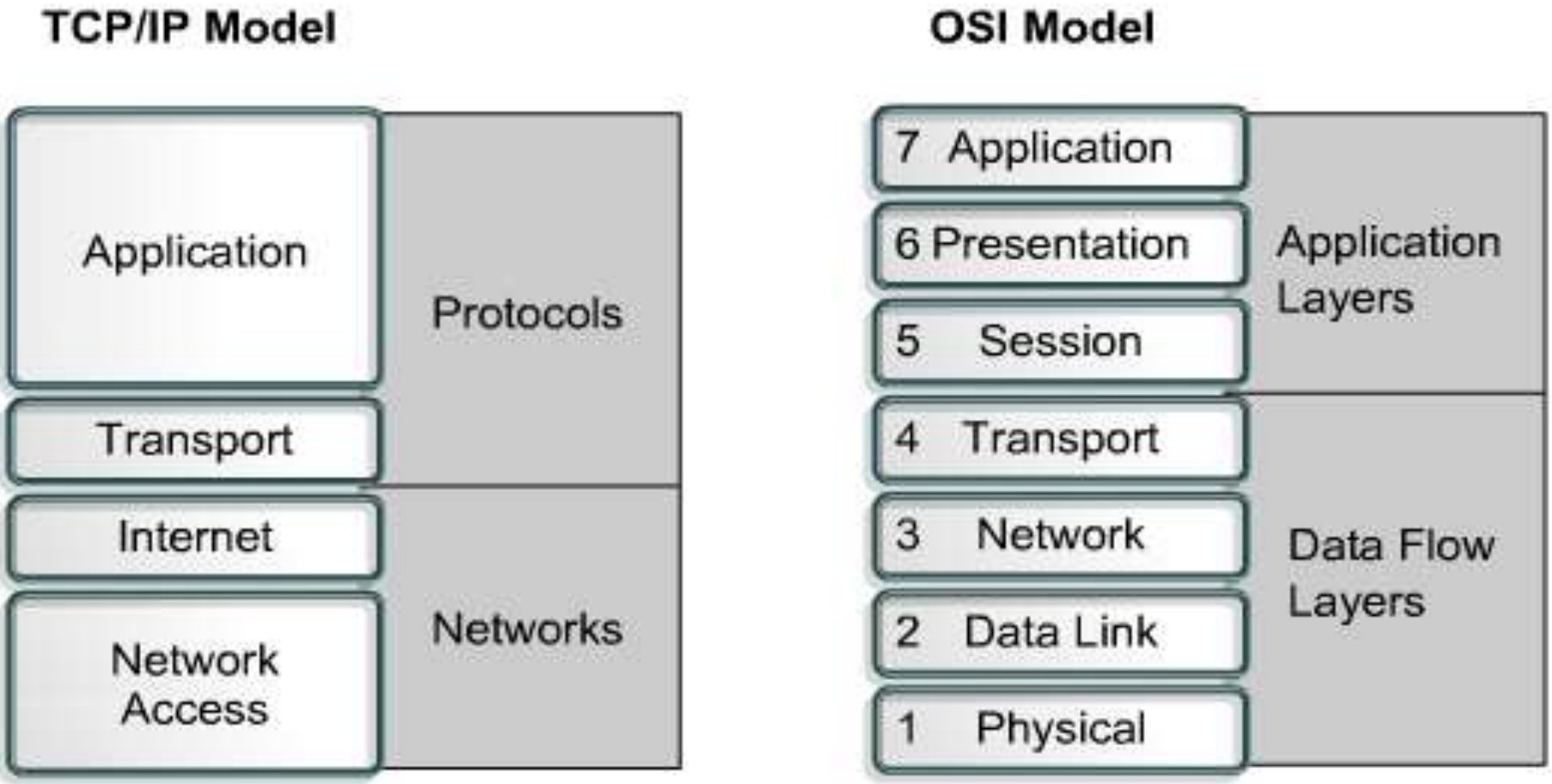
- IBM System Network Architecture (SNA)
- Digital DECnet
- Novell Netware
- Apple AppleTalk
- NetBEUI
- IPX/SPX
- TCP/IP
  - TCP (Transmission Control Protocol)
  - UDP (User Datagram Protocol)
  - IP (Internet Protocol)
  - ICMP (Internet Control Message Protocol)
  - IGMP (Internet Group Management Protocol)
  - ARP (Address Resolution Protocol)

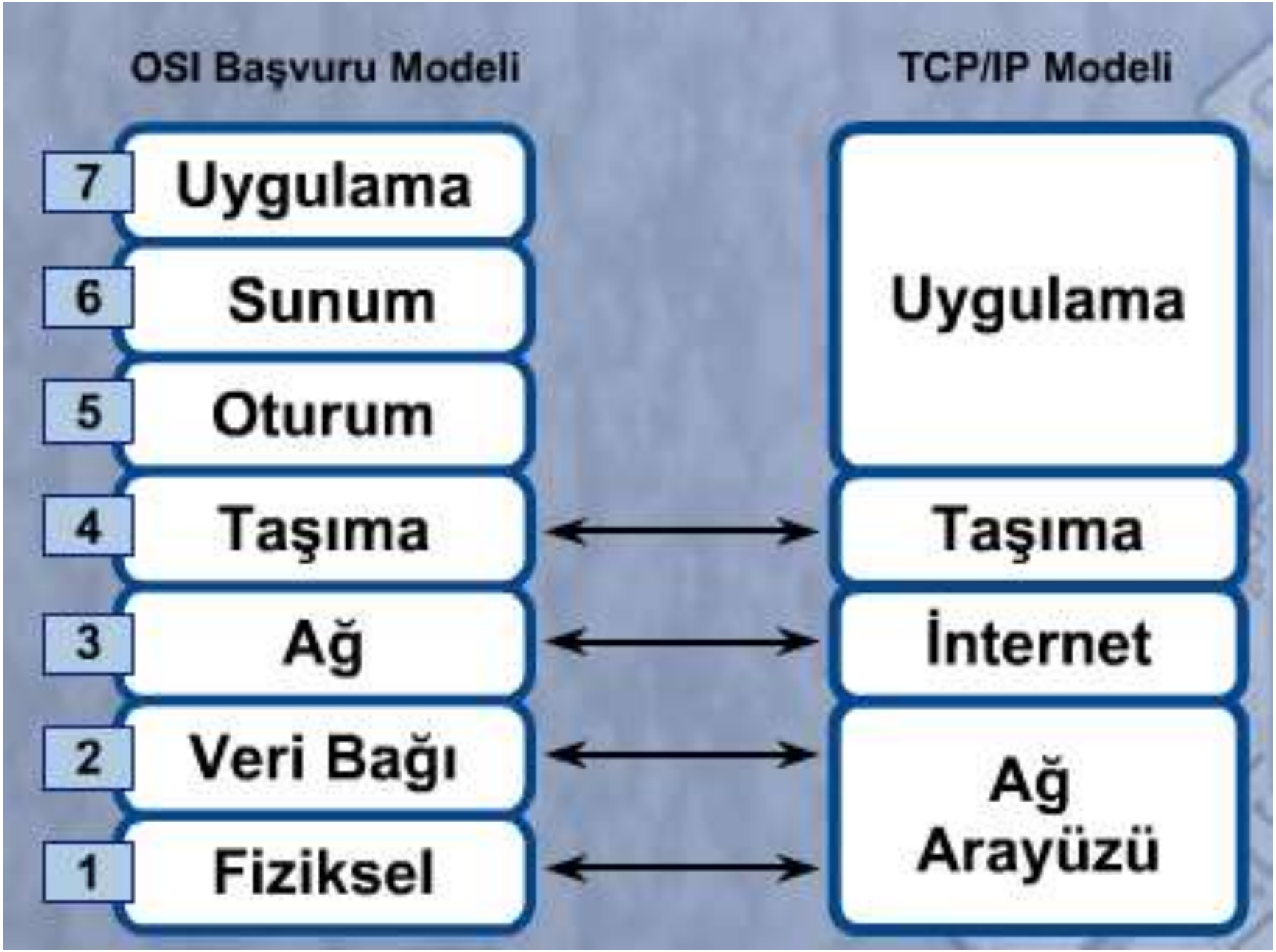
OSI Modeli


KATMANLAR
PROTOKOLLER
UYGULAMA
SUNUM
OTURUM
ULAŞIM
AĞ
VERİ
DONANIM
HTTP, HTTPS, SMTP, FTP, TFTP, UUCP, NNTP, SSL, SSH, IRC, SNMP, SIP, RTP, Telnet, ...
ISO 8822, ISO 8823, ISO 8824, ITU-T T.73, ITU-T X.409, ...
SMB, ISO 8326, NFS ISO 8327, ITU-T T.6299, ...
TCP, UDP, ...
IP, IPv4, IPv6, ICMP, ARP, IGMP, ...
Ethernet, HDLC, Wi-Fi, Token ring, ...
ISDN, RS-232, EIA-422, RS-449, EIA-485, Fiber Optik, ...

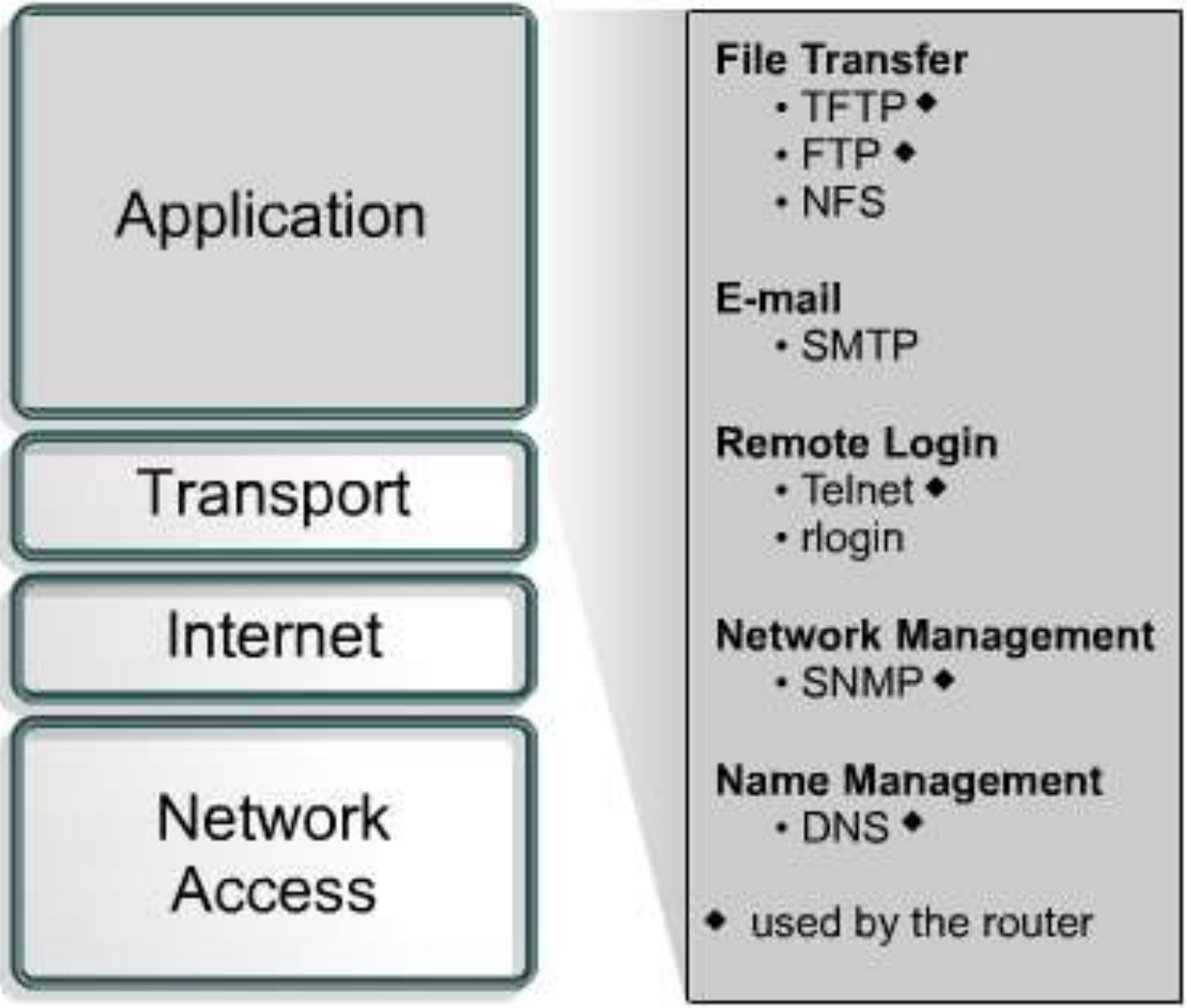

7
6
5
Application
Name System
DNS
File Sharing
NFS
Host Config
BOOTP
DHCP
Network Mgmt
SNMP
RMON
File Transfer
FTP
TFTP
E-Mail & News
RFC822 / MIME
SMTP
POP / IMAP
NNTP
WWW & Gopher
HTTP
Gopher
Inter-active
Telnet
"r" Com-mands
IRC
4
Transport
User Datagram Protocol (UDP)
Transmission Control Protocol (TCP)
3
Internet
Internet Protocol (IP/IPv4, IPv6)
IP NAT
IPSec
Mobile IP
IP Support Protocols
ICMP/ICMPv4, ICMPv6
Neighbor Discovery (ND)
IP Routing Protocols
RIP, OSPF, GGP, HELLO, IGRP, EIGRP, BGP, EGP
Address Resolution Protocol (ARP)
Reverse Address Resolution Protocol (RARP)
2
Network Interface
Serial Line Interface Protocol (SLIP)
Point-to-Point Protocol (PPP)
(LAN/WLAN/WAN Hardware Drivers)

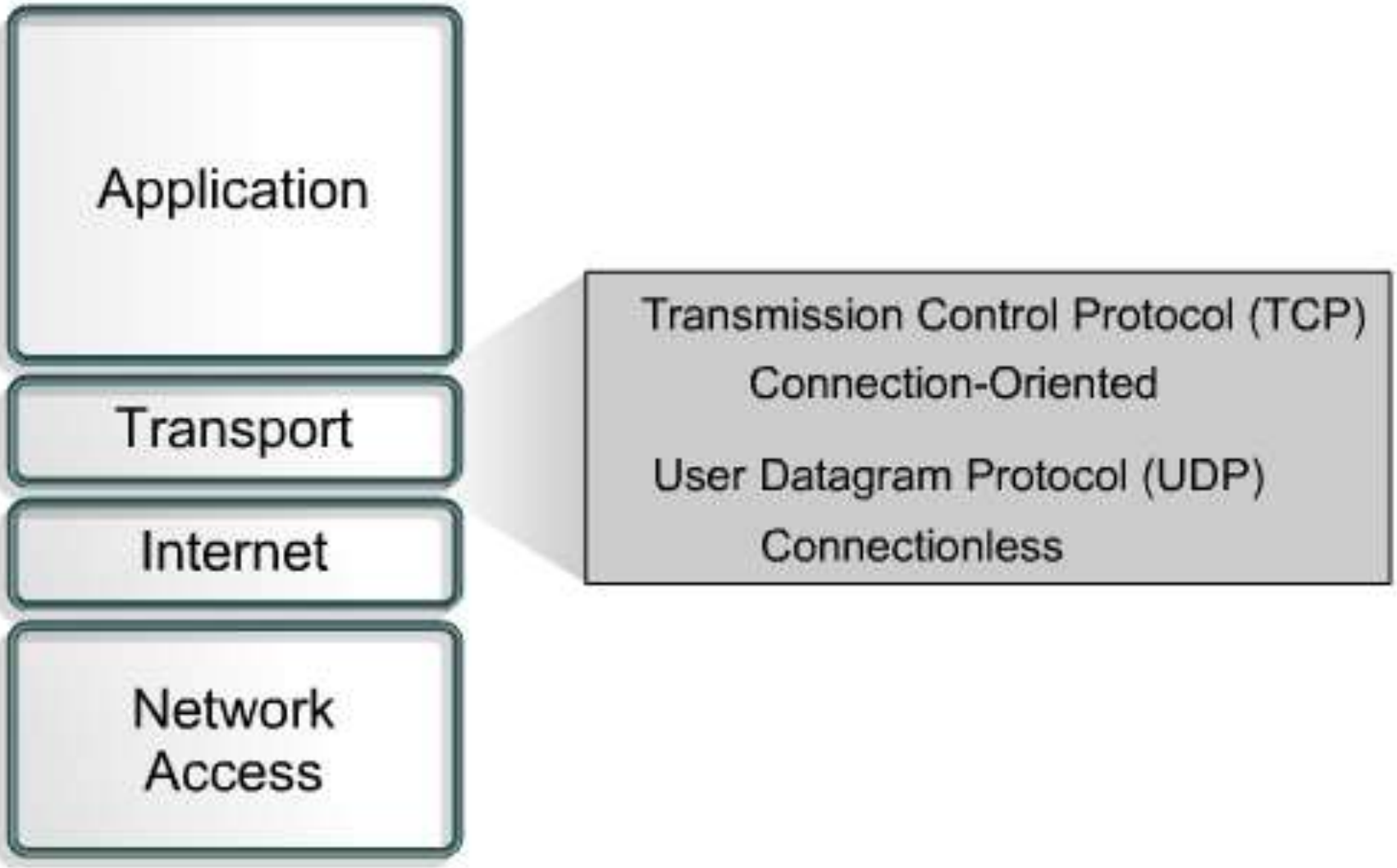

OSI Model Layers
Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data-Link Layer
Physical Layer
TCP/IP Protocol Architecture Layers
Application Layer
Host-to-Host Transport Layer
Internet Layer
Network Interface Layer
TCP/IP Protocol Suite
Telnet
FTP
SMTP
DNS
RIP
SNMP
TCP
UDP
IP
IGMP
ICMP
ARP
Ethernet
Token Ring
Frame Relay
ATM

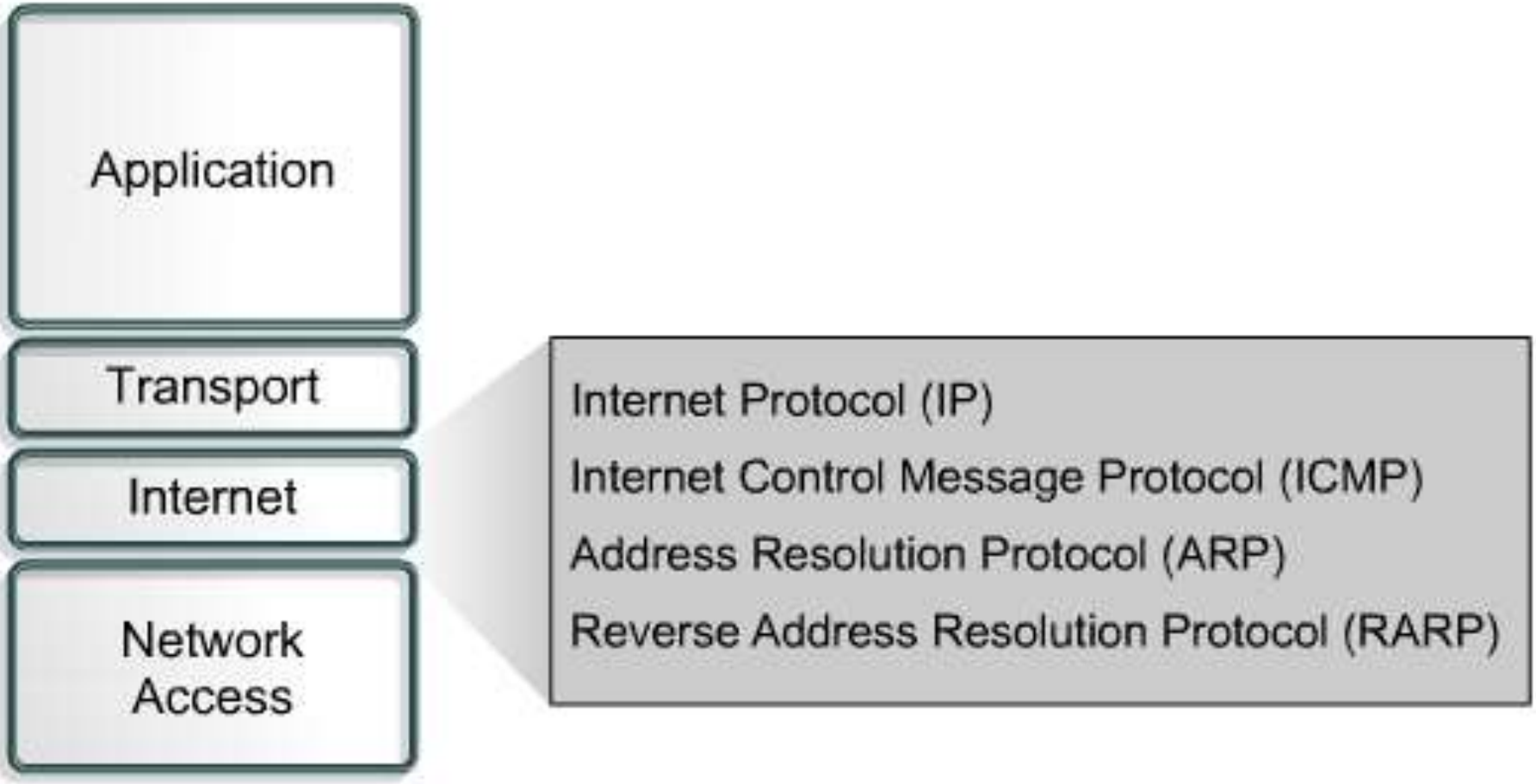

TCP/IP Model
Application
Transport
Internet
Network Access
Protocols
Networks
OSI Model
7 Application
6 Presentation
5 Session
4 Transport
3 Network
2 Data Link
1 Physical
Application Layers
Data Flow Layers

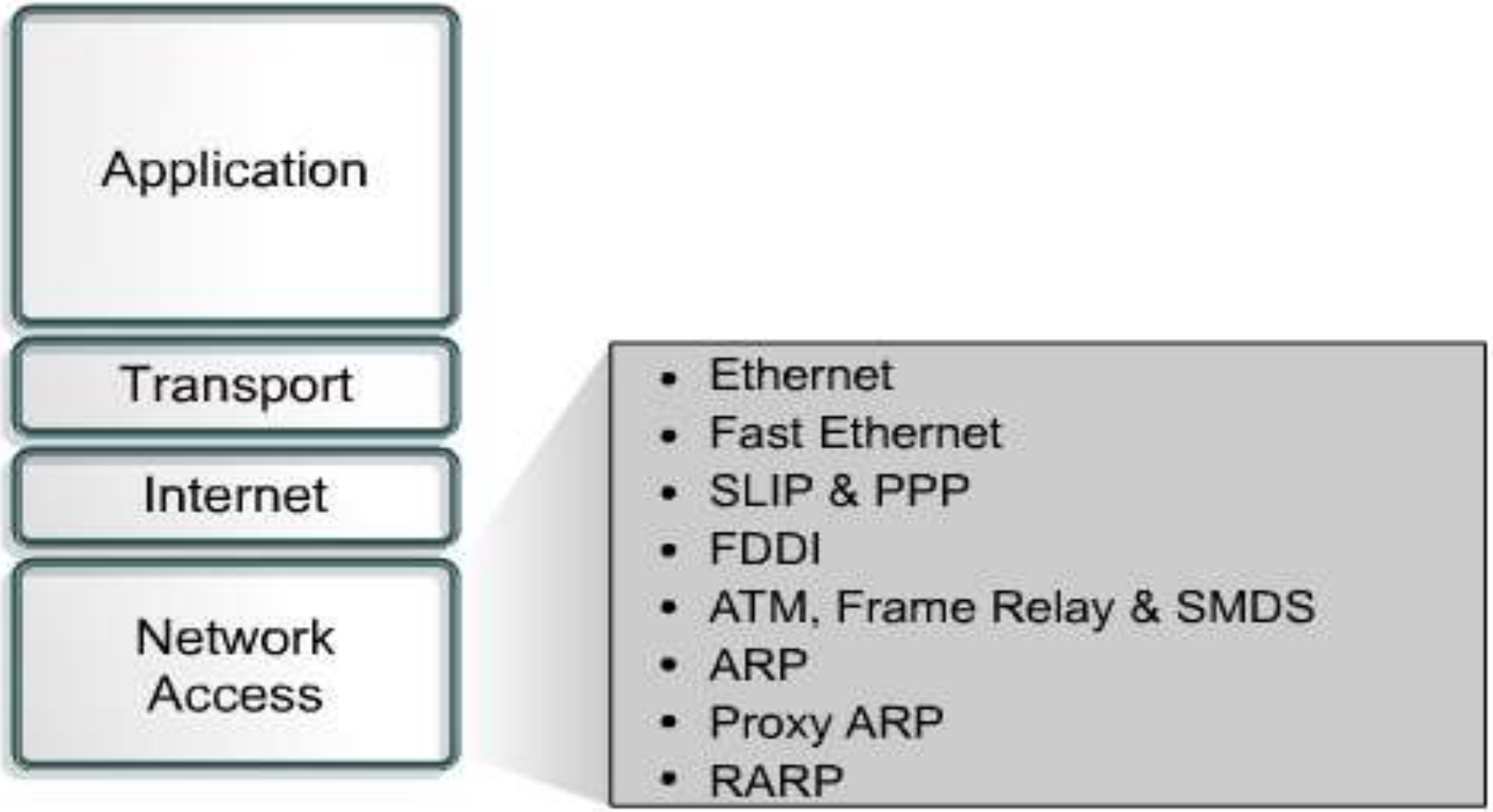

OSI Başvuru Modeli
TCP/IP Modeli
7
Uygulama
6
Sunum
5
Oturum
4
Taşıma
3
Ağ
2
Veri Bağı
1
Fiziksel
Uygulama
Taşıma
İnternet
Ağ
Arayüzü

TCP/IP Modeli	Katman	Açıklama
Uygulama	4	SMTP ve FTP gibi üst düzey protokollerin çalıştığı yer
Taşıma	3	Akış kontrolünün ve bağlantı protokollerinin olduğu yer
İnternet	2	IP adreslemenin ve yönlendirmenin yapıldığı yer
Ağ Erişimi	1	Ağda MAC adreslemenin fiziksel bileşenlerin bulunduğu yer


Application
Transport
Internet
Network Access
File Transfer
• TFTP ◆
• FTP ◆
• NFS
E-mail
• SMTP
Remote Login
• Telnet ◆
• rlogin
Network Management
• SNMP ◆
Name Management
• DNS ◆
◆ used by the router

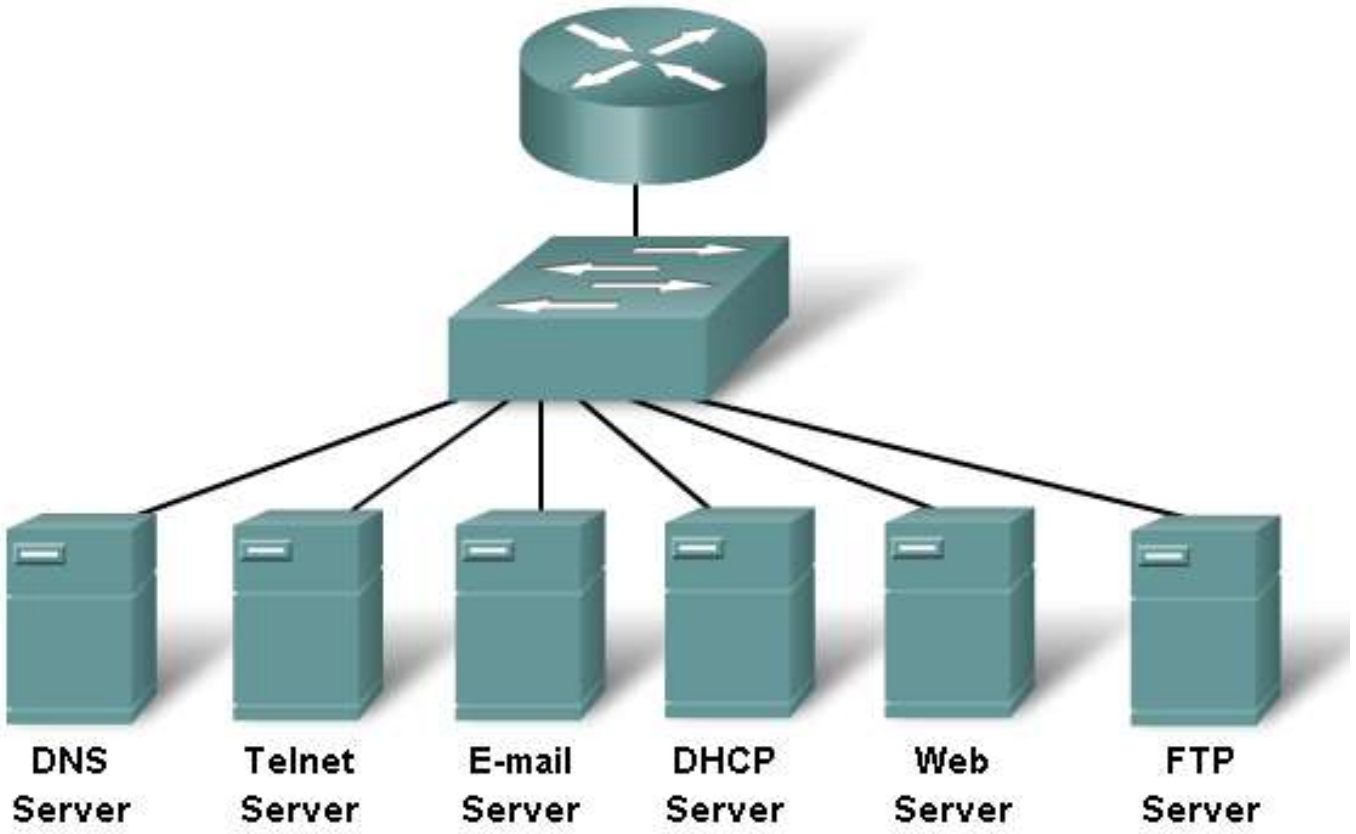

Application
Transport
Internet
Network Access
Transmission Control Protocol (TCP)
Connection-Oriented
User Datagram Protocol (UDP)
Connectionless


Application
Transport
Internet
Network Access
Internet Protocol (IP)
Internet Control Message Protocol (ICMP)
Address Resolution Protocol (ARP)
Reverse Address Resolution Protocol (RARP)

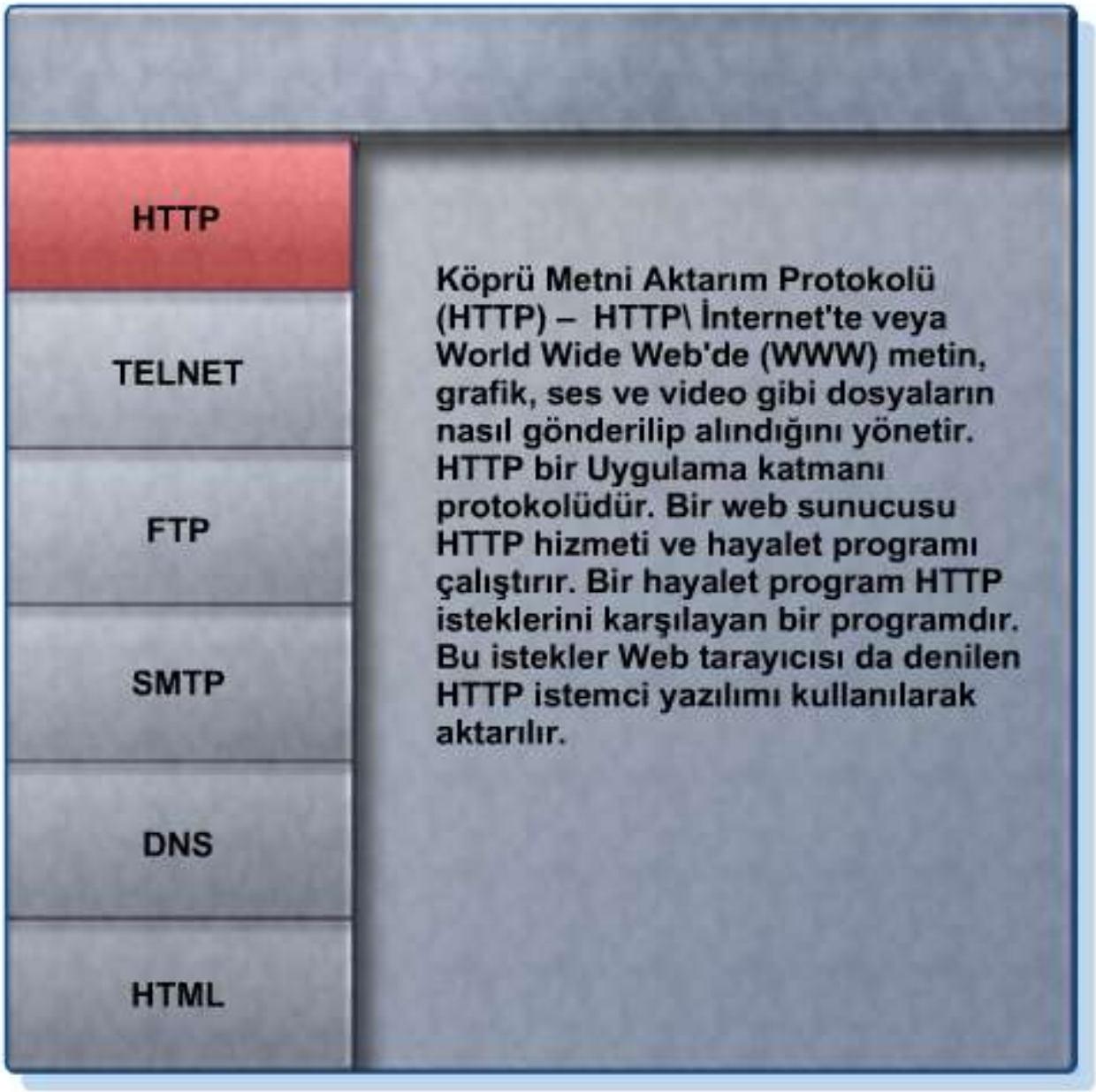

Application
Transport
Internet
Network Access
• Ethernet
• Fast Ethernet
• SLIP & PPP
• FDDI
• ATM, Frame Relay & SMDS
• ARP
• Proxy ARP
• RARP

İnternet İletişimi

- TCP / IP
- IP Adresi
- DHCP
- HTTP
- DNS
- E-mail
- FTP
- SNMP



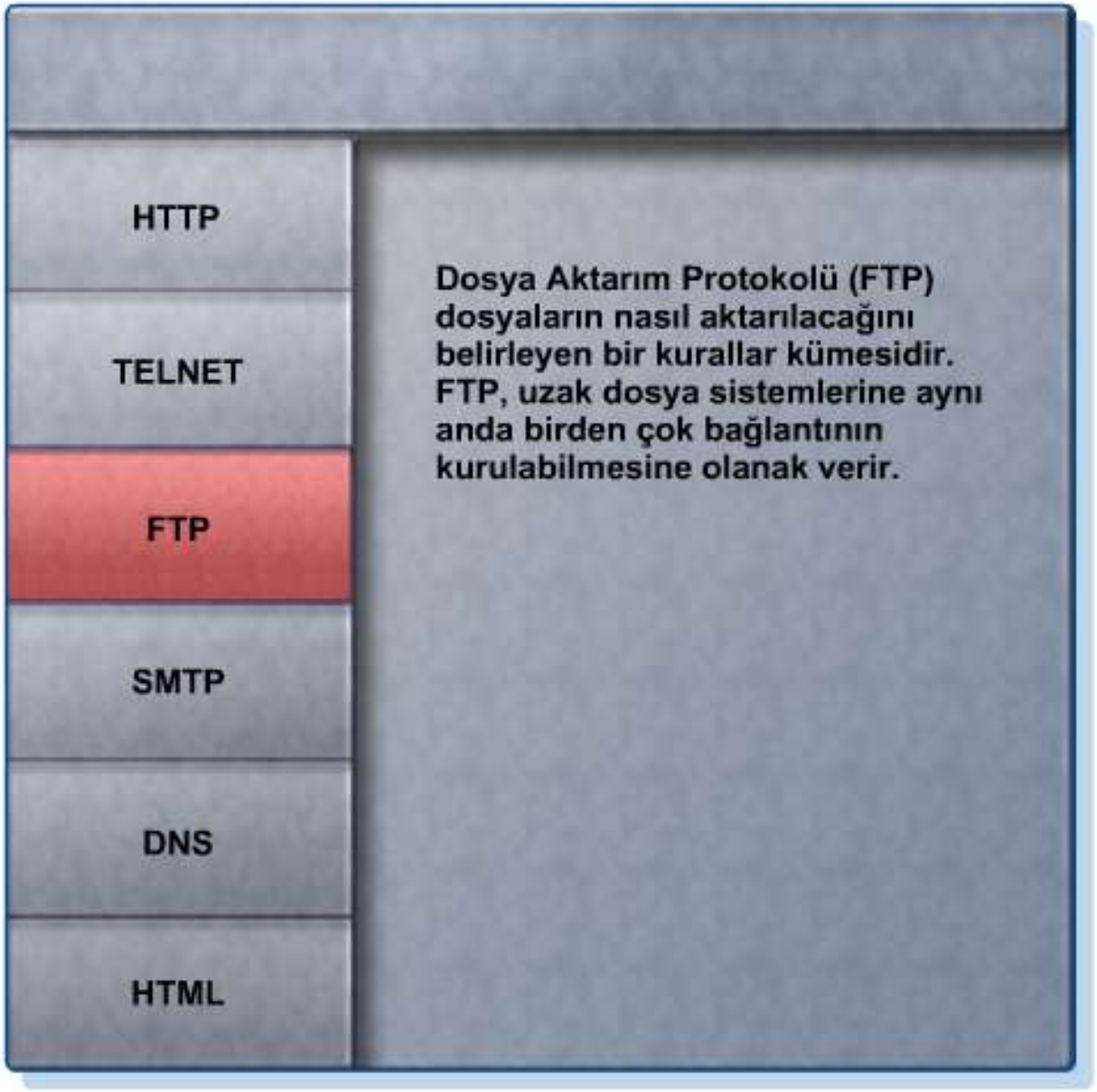
TCP/IP Uygulama Katmanı Protokolleri



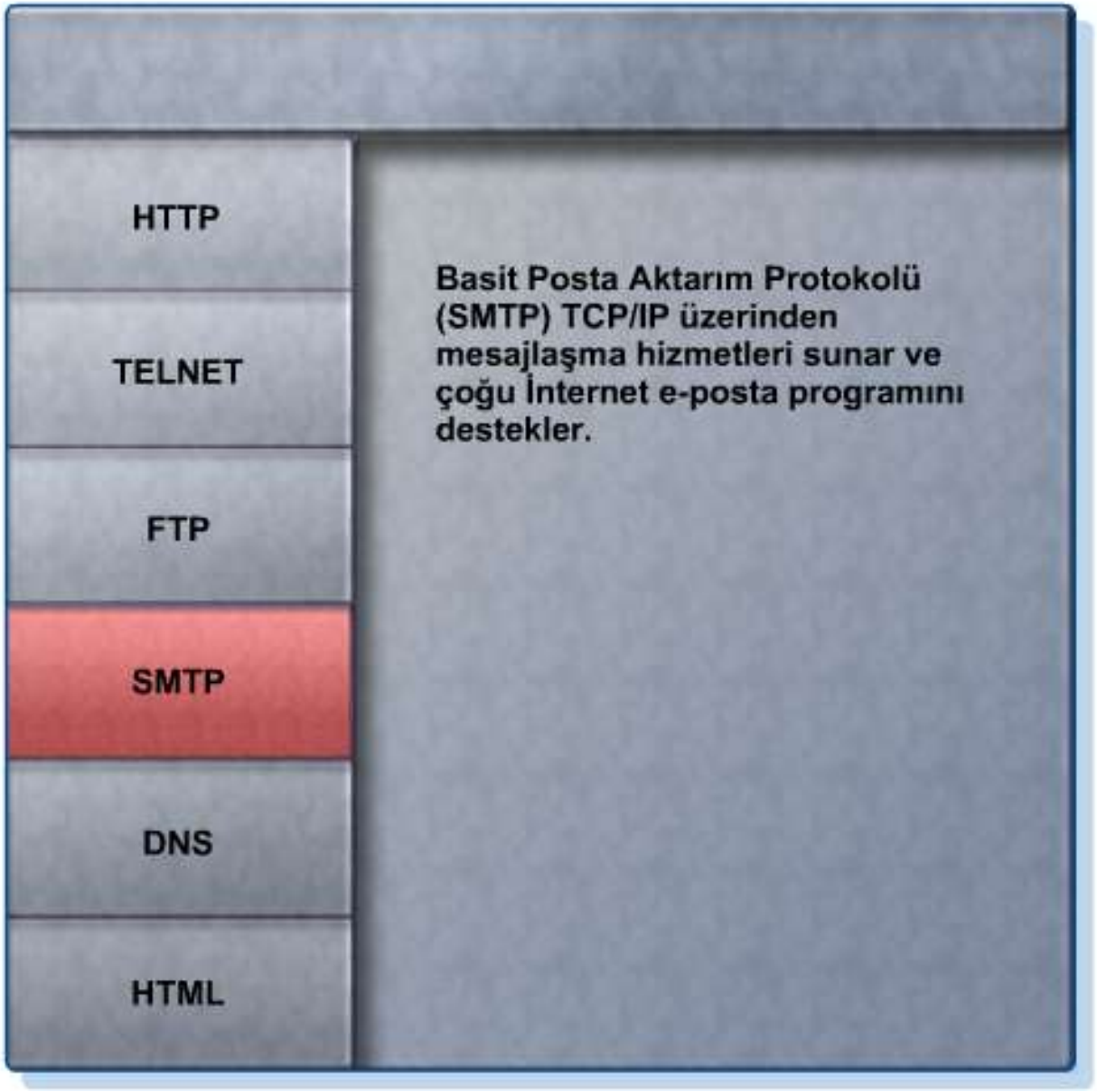
TCP/IP Uygulama Katmanı Protokolleri



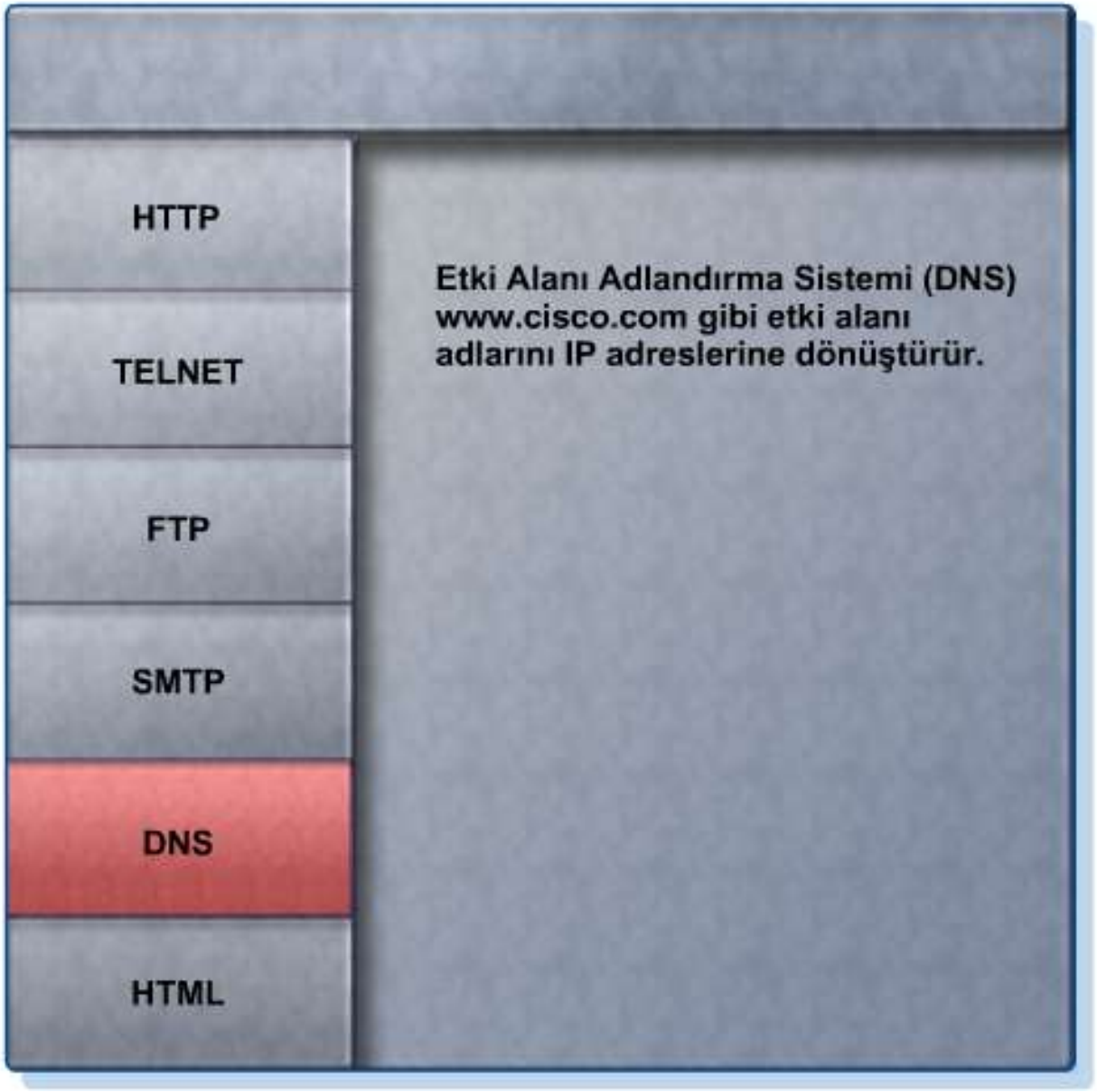
TCP/IP Uygulama Katmanı Protokolleri



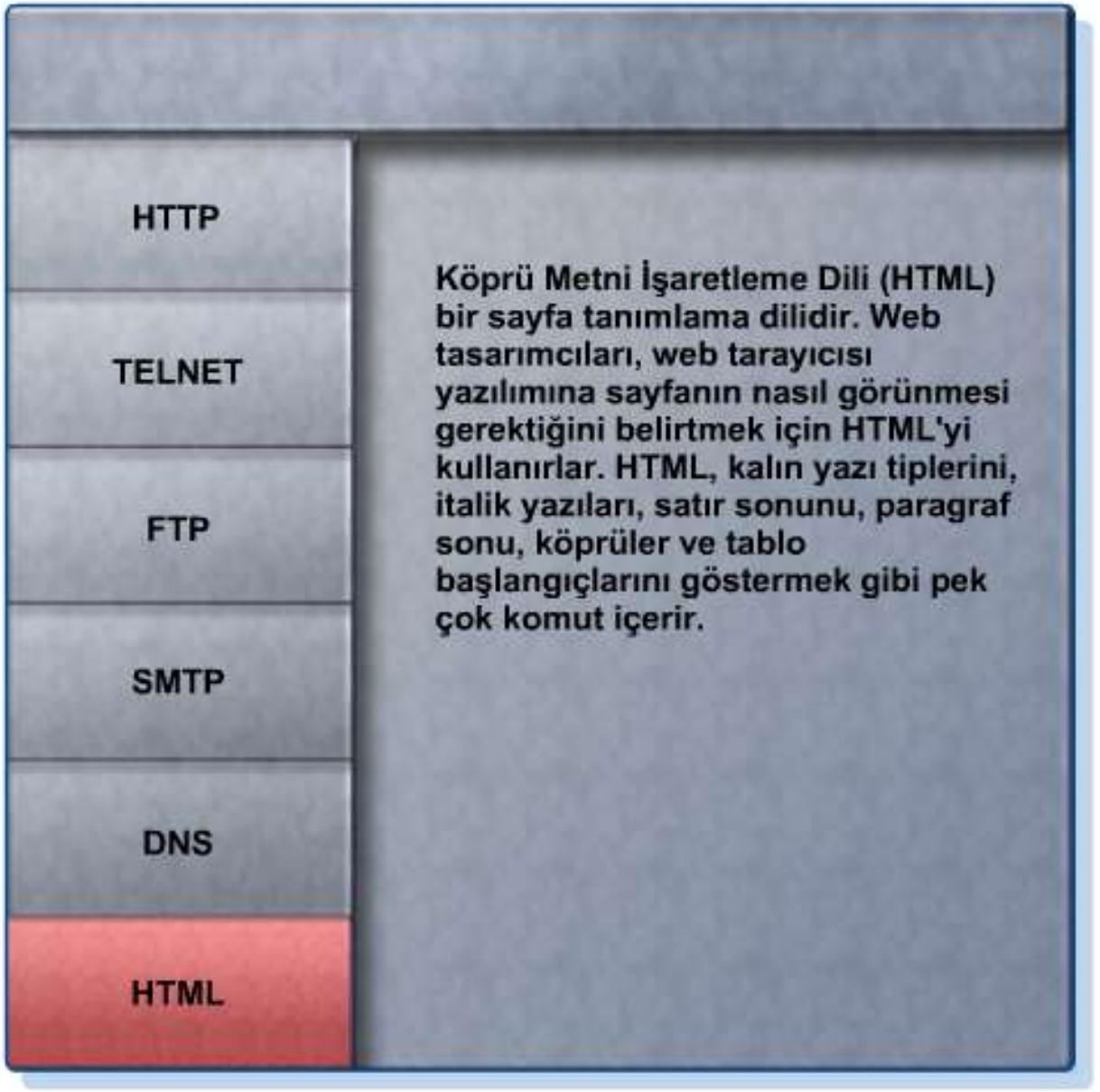
TCP/IP Uygulama Katmanı Protokolleri



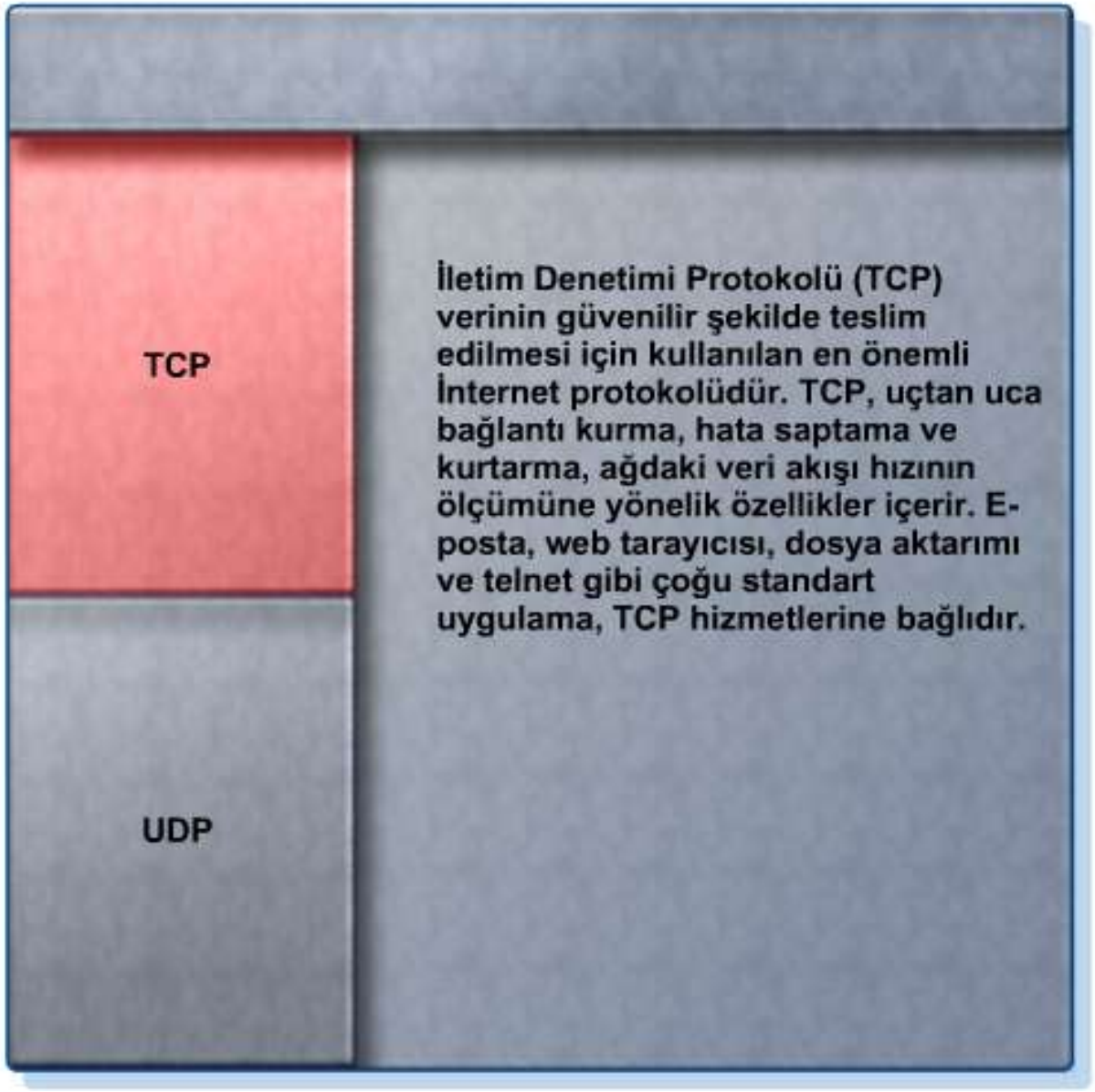
TCP/IP Uygulama Katmanı Protokolleri



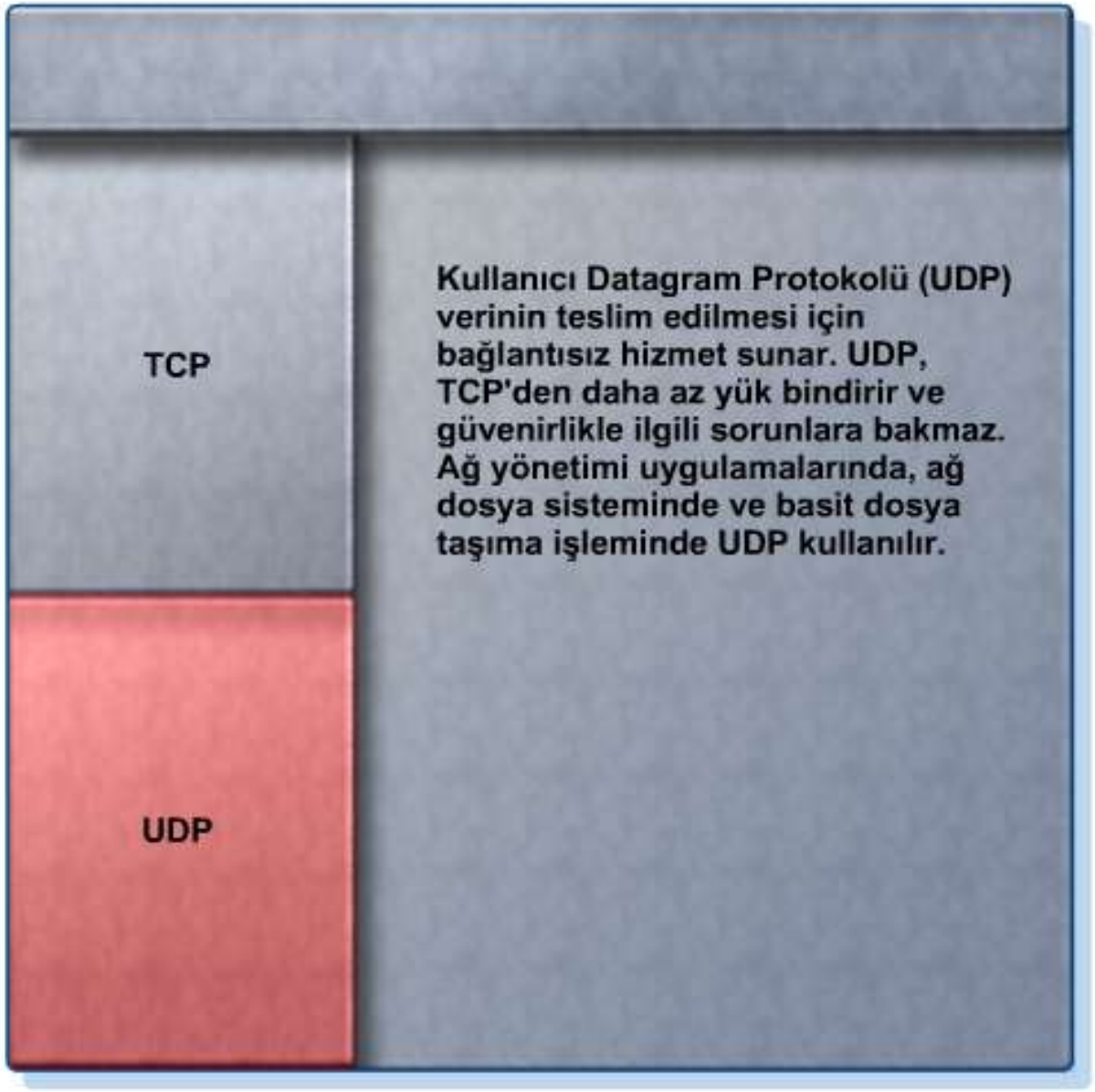
TCP/IP Uygulama Katmanı Protokolleri



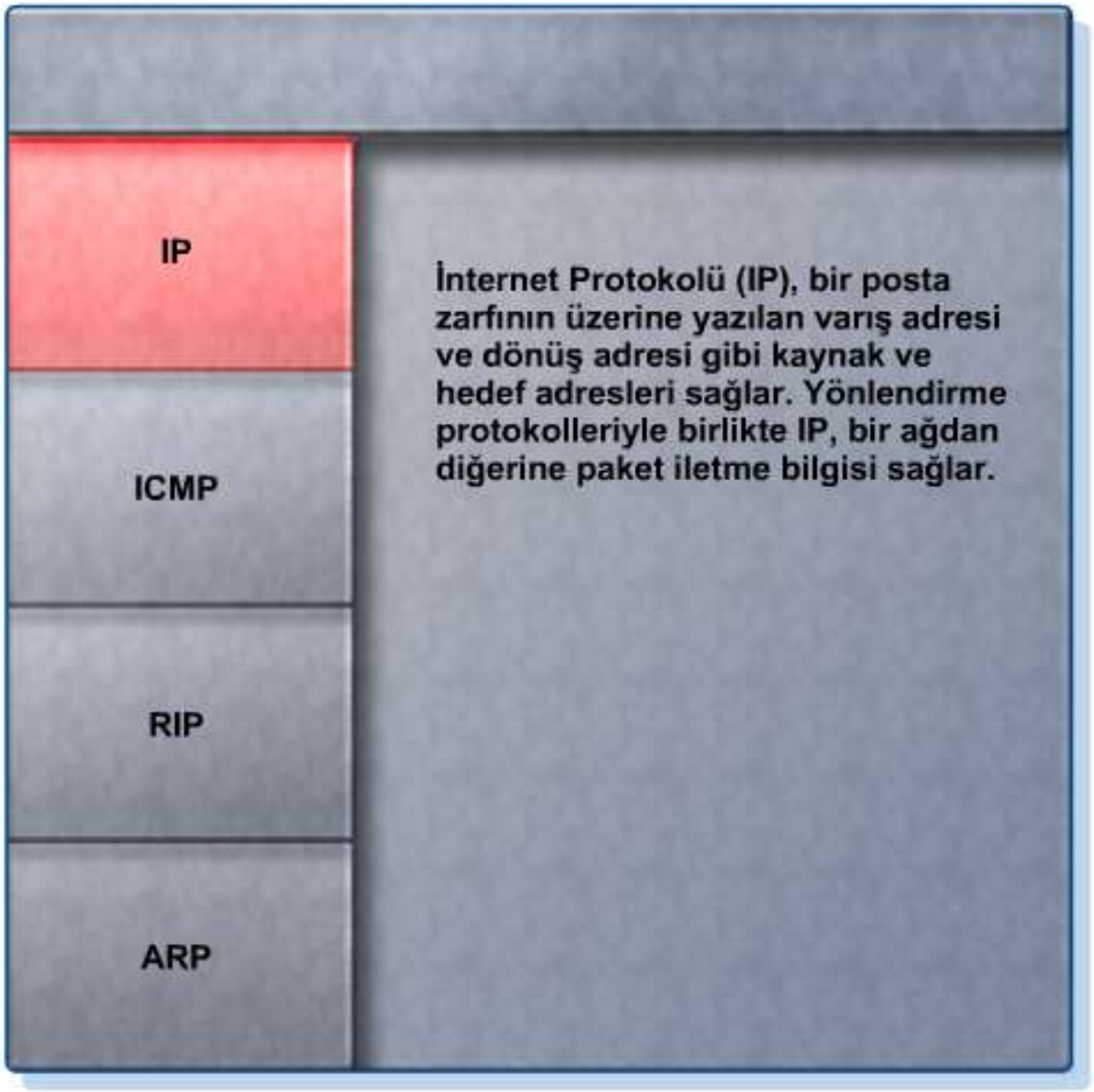
TCP/IP Taşıma Katmanı Protokolleri



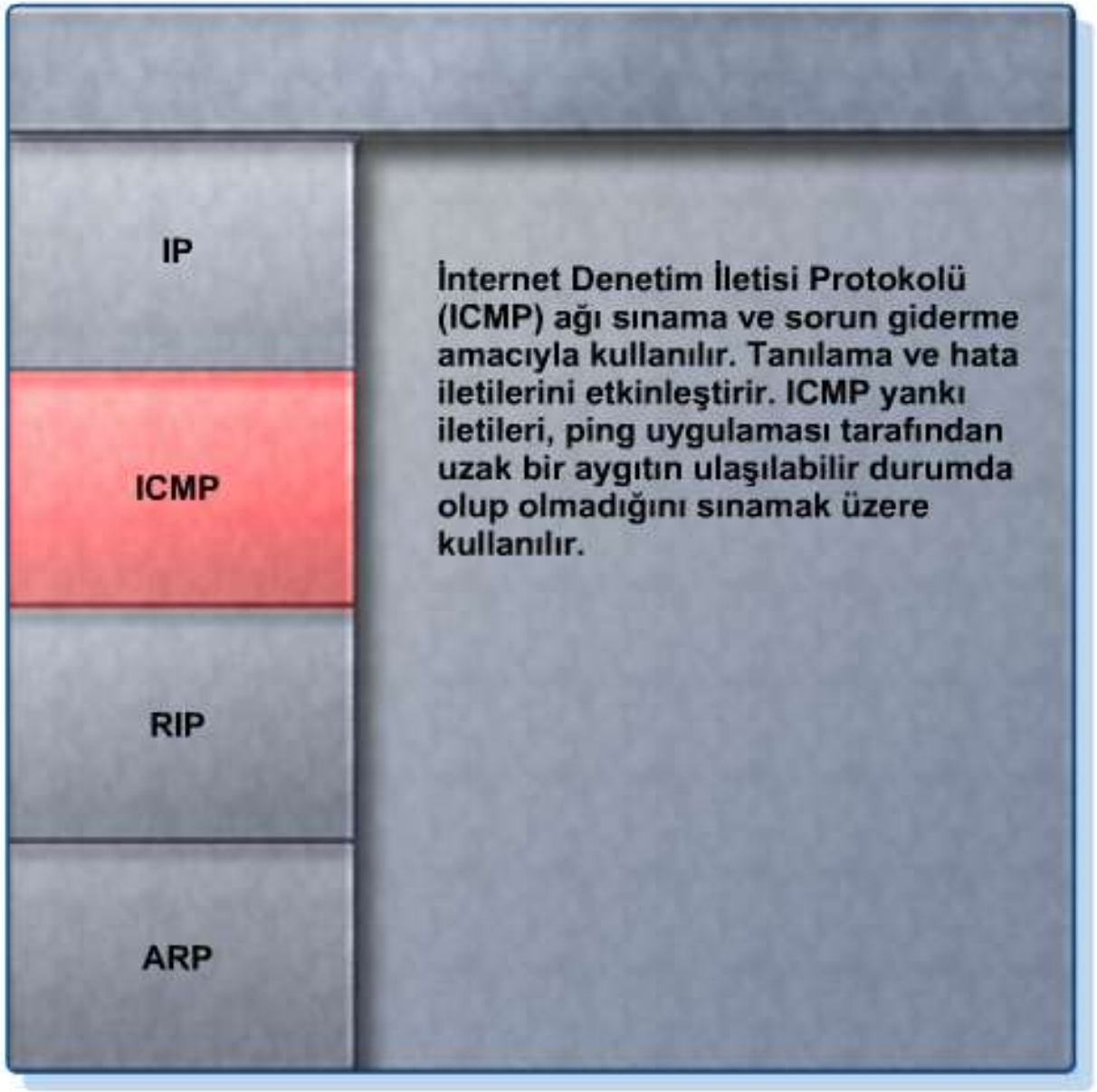
TCP/IP Taşıma Katmanı Protokolleri



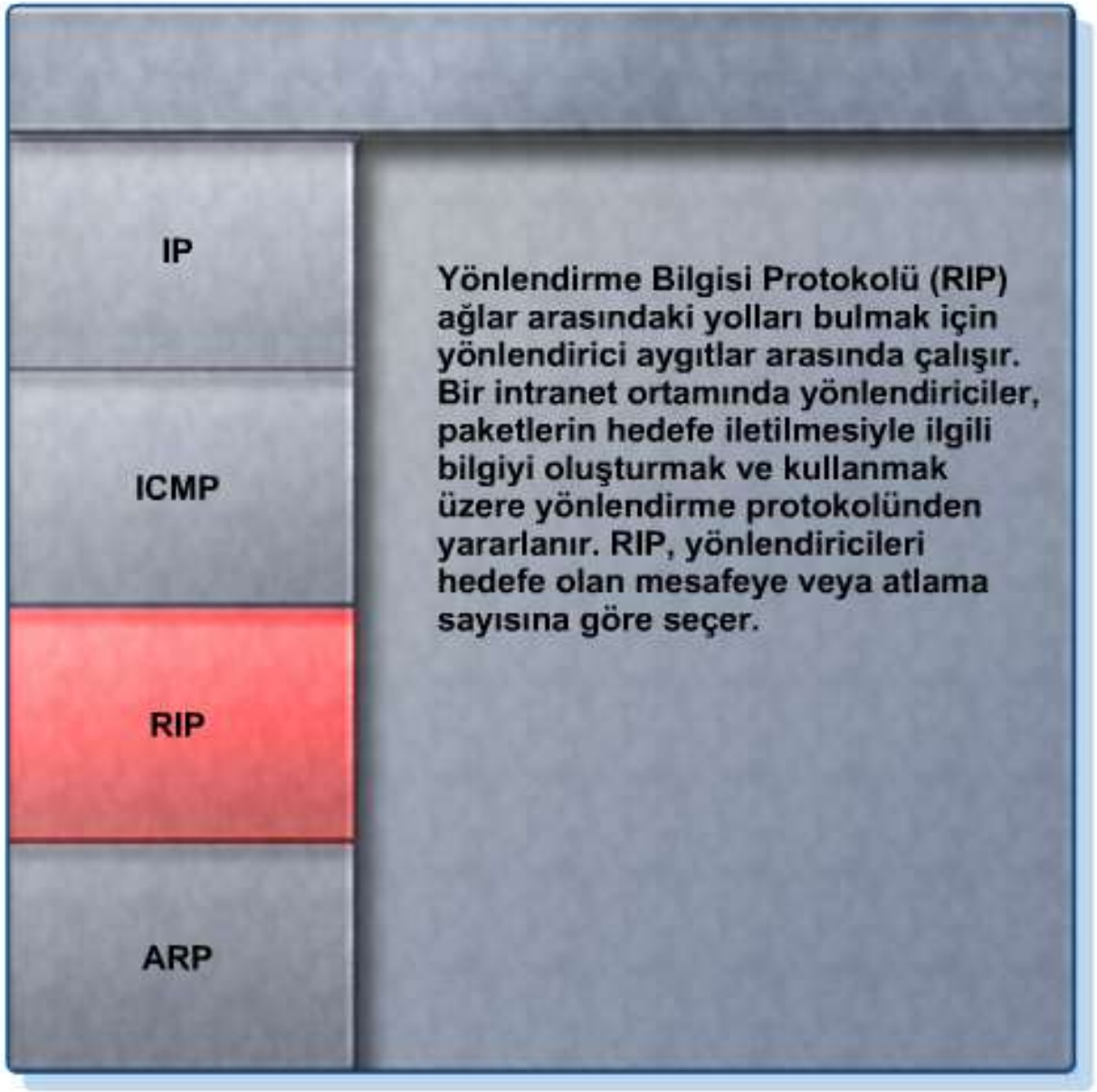
TCP/IP İnternet Katmanı Protokolleri



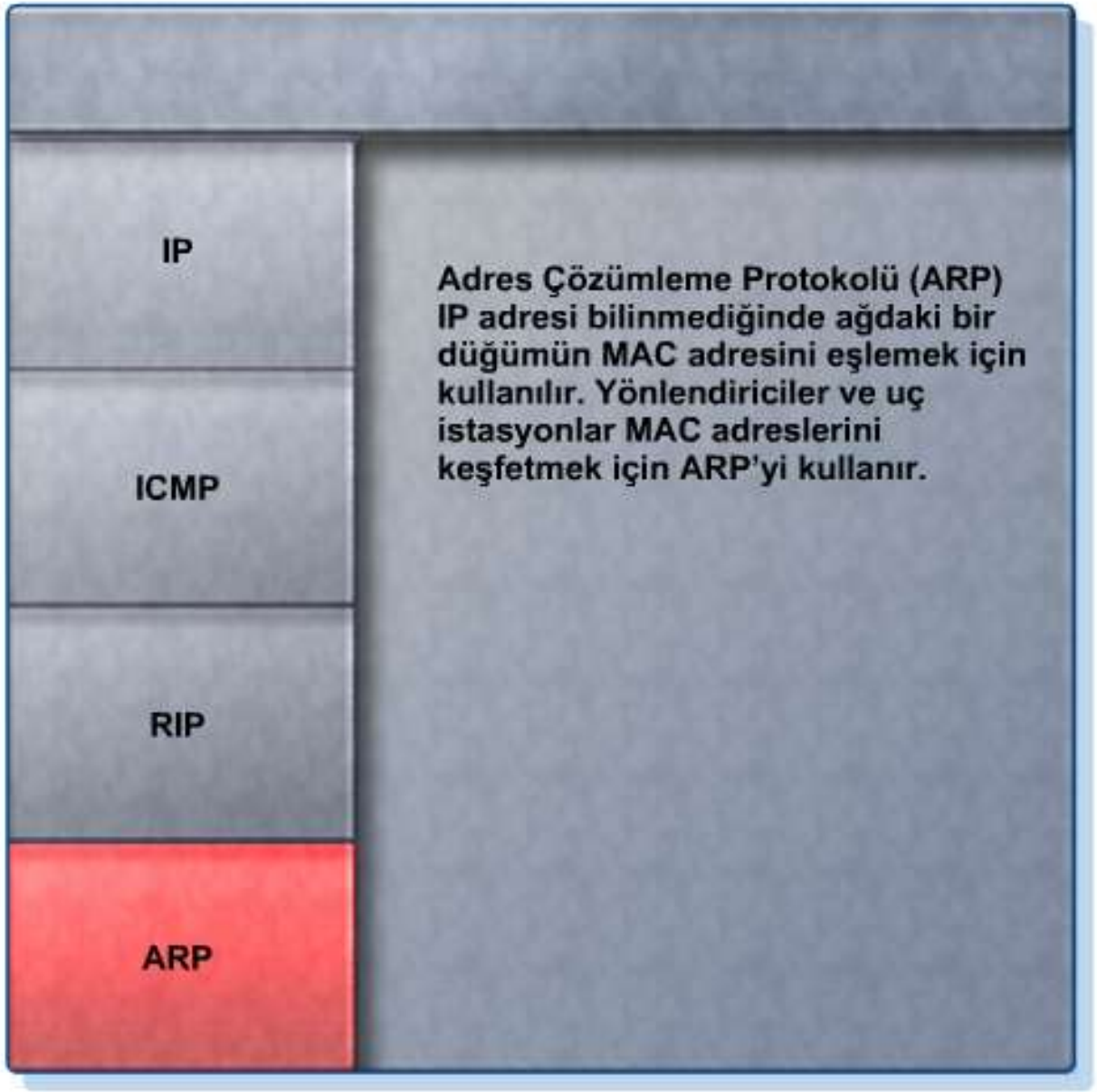
TCP/IP İnternet Katmanı Protokolleri



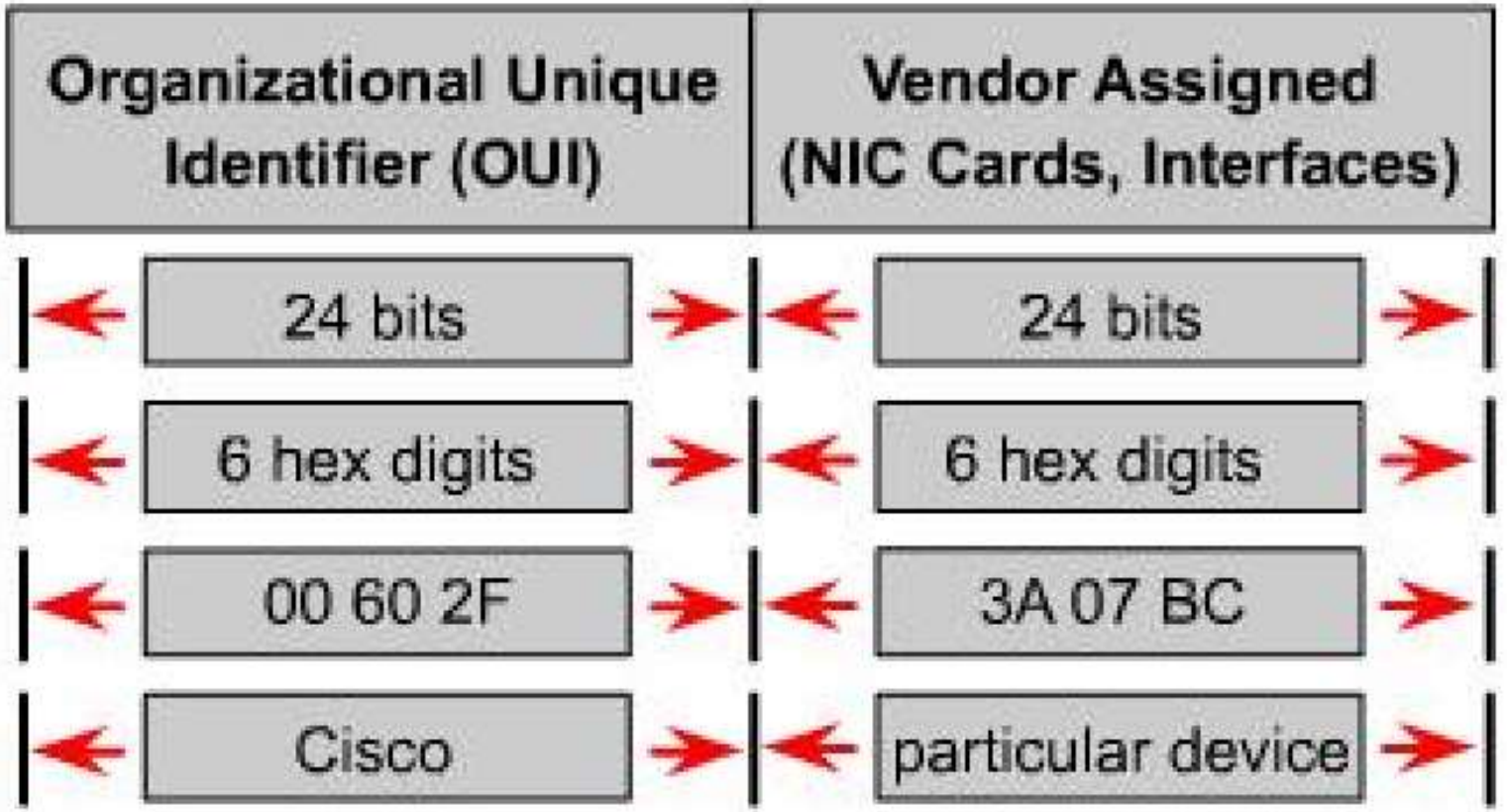
TCP/IP İnternet Katmanı Protokolleri



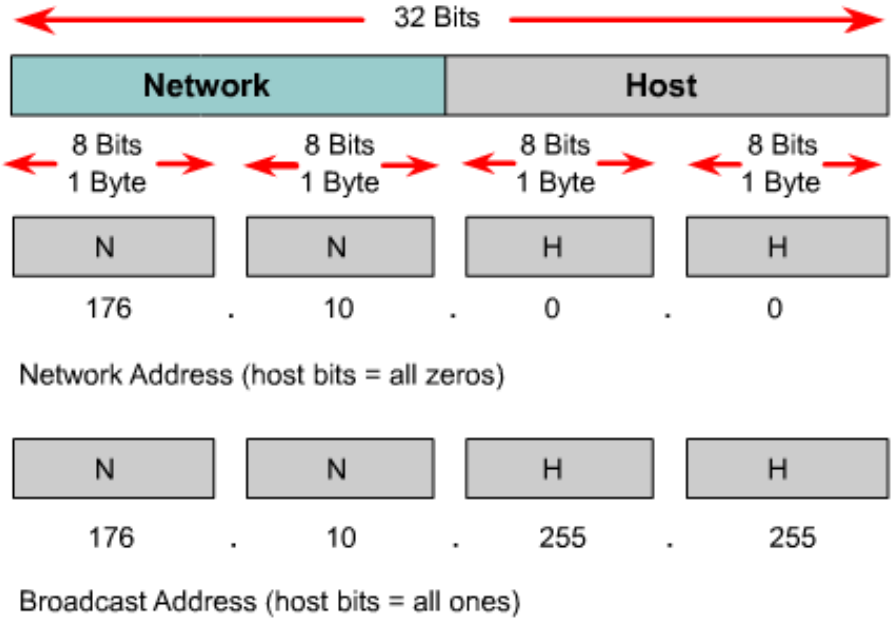
TCP/IP İnternet Katmanı Protokolleri



MAC


Organizational Unique Identifier (OUI)
Vendor Assigned (NIC Cards, Interfaces)
24 bits
24 bits
6 hex digits
6 hex digits
00 60 2F
3A 07 BC
Cisco
particular device

IP

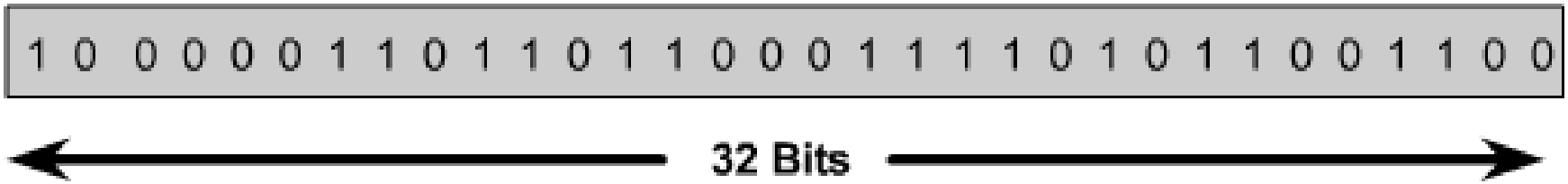


Class A	Network	Host		
Octet	1	2	3	4

Class B	Network		Host	
Octet	1	2	3	4

Class C	Network			Host
Octet	1	2	3	4

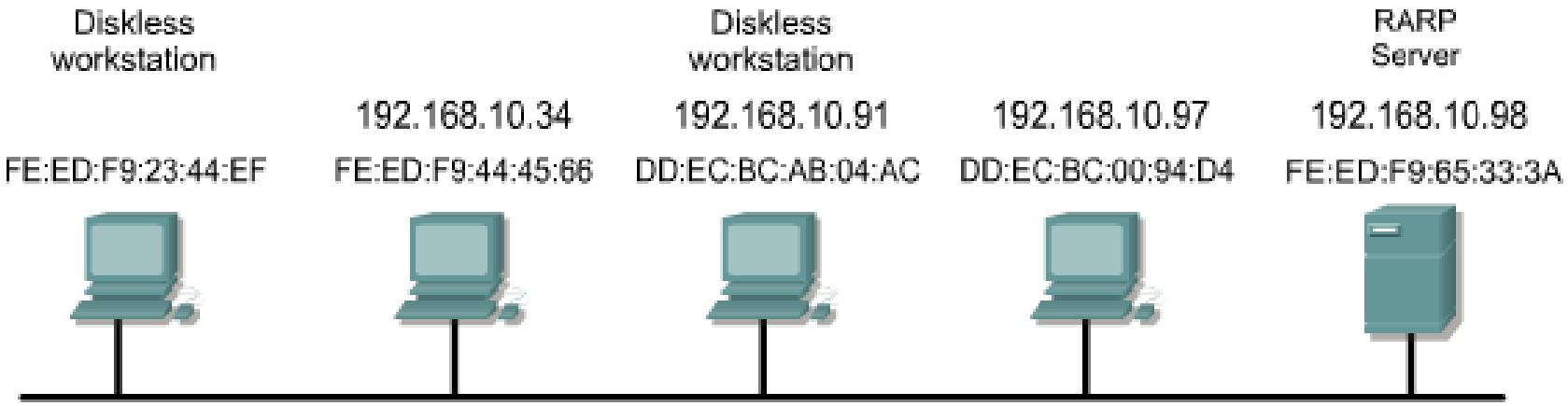
Class D	Host			
Octet	1	2	3	4



Binary : 11000000.10101000.000000001.00001000 and 11000000.10101000.00000001.00001001

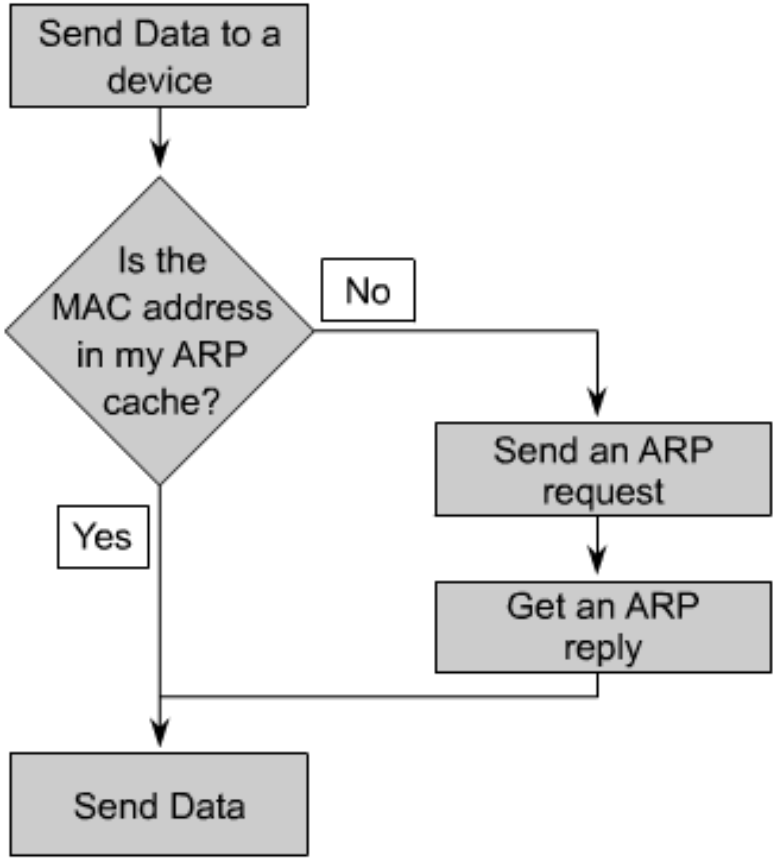
Decimal : 192.168.1.8 and 192.168.1.9

RARP



MAC HEADER	IP HEADER	RARP REQUEST MESSAGE
Destination FF-FF-FF-FF-FF-FF Source FE:ED:FD:23:44:EF	Destination 255.255.255.255 Source ????????	What is my IP address?

ARP

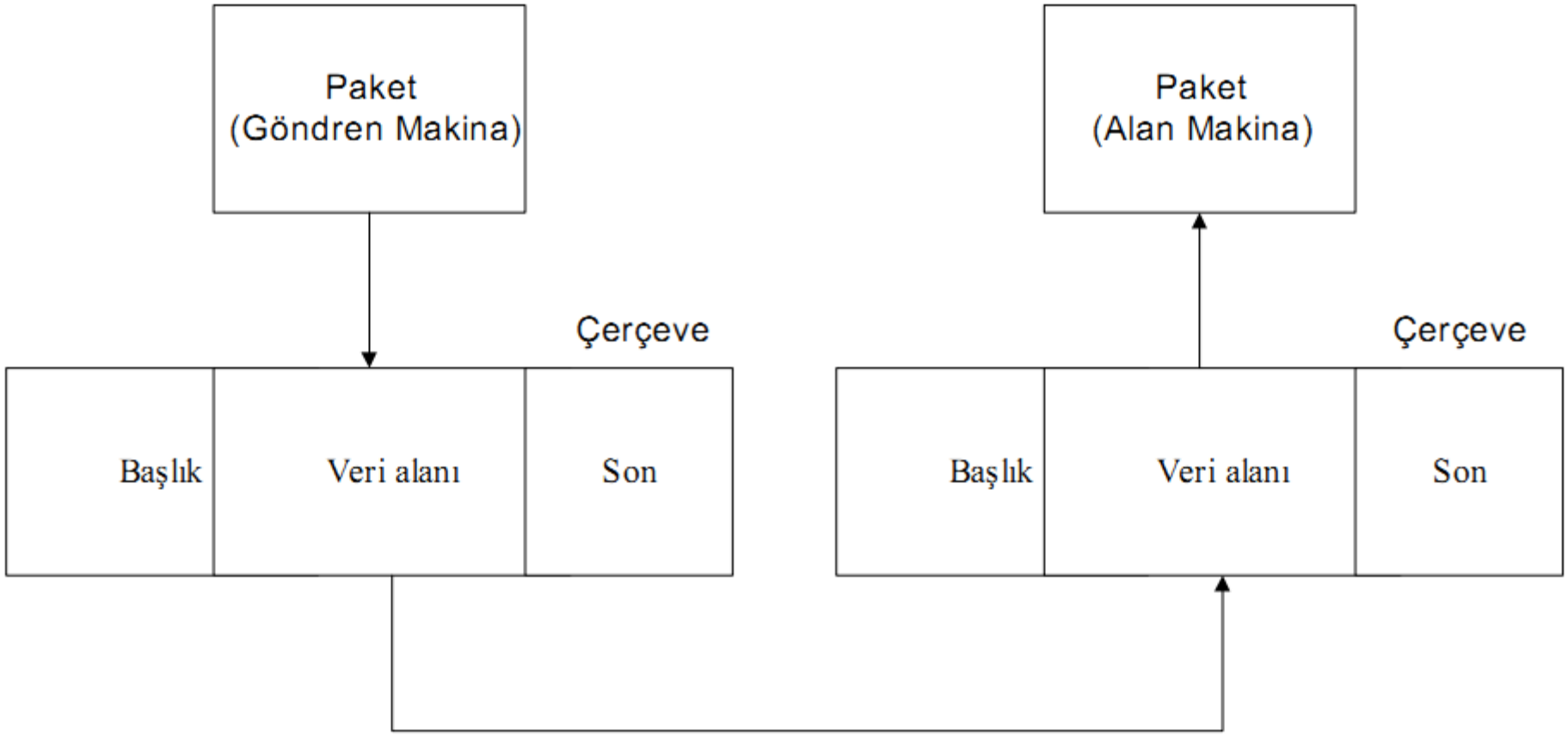

Send Data to a device
Is the MAC address in my ARP cache?
No
Yes
Send an ARP request
Get an ARP reply
Send Data

Protokol Bağlantı Noktaları

Protokol	Bağlantı Noktası	Amaç
HTTP	Bağlantı noktası 80	Web sayfalarını TCP/IP ağı üzerinde taşır
HTTPS	Bağlantı noktası 443	Web sayfalarını TCP/IP ağı üzerinde güvenli bir şekilde taşır
SMTP	Bağlantı noktası 25	TCP/IP ağı üzerinden e-posta gönderir
Telnet/SSH	Bağlantı noktası 23/22	Bilgisayarlara TCP/IP ağı üzerinden bağlantı sağlar
FTP/TFTP	Bağlantı noktası 20 veya 21	Dosyaları TCP/IP ağı üzerinde taşır
DNS	Bağlantı noktası 53	URL'leri IP adreslerine çevirir
DHCP	Bağlantı noktası 67	Bir ağ üzerinde IP adreslerinin atanmasını otomatikleştirir.

E-posta Protokollerini Karşılaştırma

Protokol	Avantajlar	Dezavantajlar	Bağlantı Noktası	Posta Gönderme	Posta Alma
SMTP	E-postayı bir sunucundan diğerine gönderir Doğrudan hedefe posta gönderebilir	Yalnızca istemci yüklemesi yapılabilir	25	E	H
POP	Basittir Kesintili bağlantıları destekler	Yalnızca indirme işlemi yapılabilir Postalar sunucu üzerinde yönetilemez	110	H	E
IMAP	Basittir POP'tan daha fazla özelliğe sahiptir Sunucuda posta depolar POP'tan daha hızlıdır Birden fazla istemcinin erişimine olanak verir	Daha fazla disk alanı ve CPU kaynağı gerektirir	143	H	E



Şekil 3.1 Veri paketi ve Çerçeve arasındaki ilişki

VIZE (ARA SINAV)

VERİ TABANI YÖNETİM SİSTEMLERİ

Temel Veri Tabanı Kavramları

Veri Nedir?

Bilgi ve veri kavramı bazen birbiri ile karıştırılmaktadır. Bilgi (information) ve veri ifadeleri çeşitli açılardan açıklanabilir. Genellikle veri ile bilgi arasında farklılık olduğu ve bilgiyi elde etmeye yarayan işlenmemiş ham malzemenin veri olduğu kabul edilir. Öğrenilmek istenilen şeyleri bildikten ve veriyi kullanmaya başladıktan sonra bilgi ifadesi ortaya çıkar. Bilgi şimdi bilinen ve gelecek zamanda verilecek olan kararlar için var olan gerçek bir değerdir ve anlamlı biçimde derlenen ve birleştirilen verilerden oluşur. Bir kaynaktan, bir alıcıya iletilen mesajın içeriğidir ve bu anlamda bilgi, karar verme ile bağlantılıdır ve veriye göre daha etkin bir kavramdır.

Kısaca veri bilgiyi elde etmeye yönelik olan ve işlenmemiş ham bir malzeme, dağınık haldeki bir topluluktur. Bilgi ise verilerden anlamlı bir bütün teşkil edecek şekilde bir araya getirilmiş ve gelecekle ilgili olayları yada organizasyonları etkileyecek bir topluluktur.

Veri Tabanı Nedir?

Veri tabanı (Database), birbirleriyle ilişkisi olan verilerin tutulduğu, kullanım amacına uygun olarak düzenlenmiş veriler topluluğunun mantıksal ve fiziksel olarak tanımlarının olduğu ve bunların sayısal ortamlarda saklandığı ve gerektiğinde tekrar bir erişime olanak sağlayan, büyük boyutlarda veriler barındıran bilgi depolarıdır. Veri tabanları gerçekte var olan ve birbirleriyle ilişkileri olan nesneleri ve ilişkilerini modeller. Veri tabanı; banka, üniversite, okul, seyahat şirketi, hastane vb kuruluşların çalışıp işleyebilmesi için gereken uygulama programlarının kullandığı operasyonel çok çeşitli verilerin toplamıdır. Ticari bir şirket için müşteri bilgileri, satış bilgileri, ürün bilgileri, ödeme bilgileri, vb., okul için öğrenci bilgileri, açılan dersler, okula kaydedilmiş öğrenciler, öğretmen bilgileri, sınav tarihleri, sınav sonuçları vb., hastane için hasta bilgileri, doktor bilgileri, teşhis-tedavi bilgileri, mali bilgiler vb. kullanılan çok çeşitli operasyonel verilere örnek olarak verilebilir.

Belirli bir konu hakkında toplanmış veriler bir veritabanı programı altında toplanırlar. Bu verilerden istenildiğinde; toplanan bilgilerin tümü veya istenilen özelliklere uyanları görüntülenebilir, yazdırılabilir ve hatta bu bilgilerden yeni bilgiler üretilerek bunlar çeşitli amaçla kullanılabilir.

Veri tabanı yönetim sistemi(VTYS), yeni veritabanları oluşturmak, veri tabanını düzenlemek, geliştirmek ve bakımını yapmak gibi çeşitli karmaşık işlemlerin gerçekleştirildiği birden fazla programdan oluşmuş bir yazılım sistemidir. Veri tabanı yönetim sistemi, kullanıcı ile veri tabanı arasında bir arabirim oluşturur ve veri tabanına her türlü erişimi sağlar. Veri tabanı ile ilgili bazı tanımlar aşağıda listelenmiştir. Bunlar:

Veri Tabanı Sistemi: veri tabanlarını kurmayı, oluşturmayı, tanımlamayı, işletmeyi ve kullanmayı sağlayan programlar topluluğudur. Veri tabanı yönetim sistemi(VTYS) veya Database Management System(DBMS) olarak ta bilinir. Eğer söz konusu veri tabanları yapısı ilişkisel yapıda ise, veri tabanı sistemi, ilişkisel veri tabanı sistemi olarak adlandırılır. (Relational Database Management System).

Veri tabanının tanımlanması: Veri tabanını oluşturan verilerin tip ve uzunluklarının belirlenmesidir.

Veri tabanının oluşturulması: Veri için yer belirlenmesi ve saklama ortamına verilerin yüklenmesini ifade eder.

Veri tabanı üzerinde işlem yapmak: Belirli bir veri üzerinde sorgulama yapma, meydana gelen değişiklikleri yansıtmak için veri tabanının güncellenmesi ve rapor üretilmesi gibi işlemleri temsil eder.

Verinin bakım ve sürekliliği: Veri tabanına yeni kayıt eklemek, eskileri çağırmak ve gerekli düzenleme, düzeltme ve silme işlemlerini yapmak gibi işlemlerin gerçekleştirilmesini ifade eder. Veri tabanı yönetim sistemi aynı zamanda verinin geri çağrılabilmesini de sağlar.

Veri tabanını genişletme: Kayıtlara yeni veri eklemek ve yeni kayıtlar oluşturmak gibi işlemleri ifade eder.

Bir veri tabanından beklenen özellikler, verileri koruması, onlara erişilmesini sağlaması ve başka verilerle ilişkilendirilmesi gibi temel işlemleri yapabilmesidir. Veri tabanı kullanarak verilerden daha kolay yararlanılabilir, istenilen verilere çok kolay erişilebilir, çeşitli sorunların çözümünde yardımcı olacak yeni bilgiler üretilebilir.

En önemlisi veriler bir merkezde toplanabilir, herkesin bu verilere yetkileri ölçüsünde erişmesi, düzeltmesi, silmesi veya görmesi sağlanabilir. Böylece veri girişinde ve veriye erişimde etkinlik ve güvenirlilik sağlanır.

Veri tabanı kullanıldığı zaman, bir kurulaşa ait tüm operasyonel veriler merkezi bir yerde ve merkezi kontrol altında tutulmuş olur.

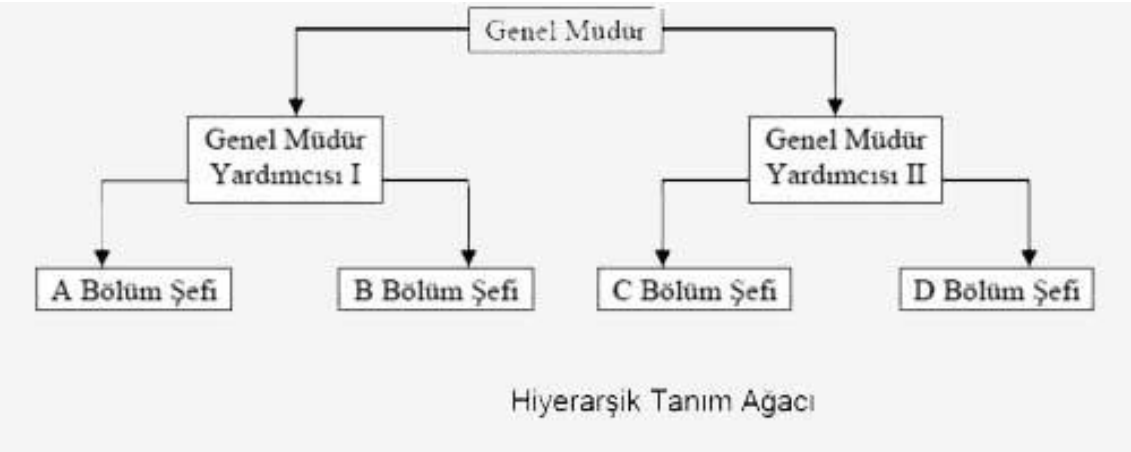
Bir veritabanı üzerinde birden fazla veri tabanı bileşeni vardır; bu bileşenler, saklanmak istenen ham bilginin belli bir formatta alınarak, veri haline gelmesi işleminde etkin rol oynarlar.

Veri tabanı yönetim sistemi programları, fiziksel hafızayı ve veri tiplerini kullanıcılar adına şekillendirip denetleyen ve kullanıcılarına standart bir SQL arayüzü sağlayarak onların dosya yapıları, veri yapısı, fiziksel hafıza gibi sorunlarla ilgilenmek yerine veri giriş-çıkışı için uygun arayüzler geliştirmelerine olanak sağlayan yazılımlardır. VTYS'de kullanıcılar, roller ve gruplar vardır ve bunlar verileri tutmak üzere bir çok türde nesne ve bu nesnelere erişimleri düzenleme görevi yaparlar. Her bir kullanıcının veri tabanı yöneticisi tarafından yapılan tanımlanmış belirli hakları vardır. Bu haklar verilebilir, verilmiş haklar artırılabilir, kısıtlanabilir veya silinebilir. Örneğin bir tablo ya da programı bir kullanıcı kullanabiliyorken bir başkasının hakları kaldırılabilir veya kısıtlanabilir.

Veri tabanı yönetim sistemlerinin sınıflandırılması

Veri modeline göre sınıflandırma:

Hiyerarşik Veri Tabanları : veri tabanları için kullanılan ilk modeldir. Bu veri tabanı tipi, ana bilgisayar ortamında çalışan yazılımlar tarafından kullanılmaktadır. IBM tarafından çıkarılan IMS bu türde en çok kullanılan yazılımdır. Hiyerarşik model, bir ağaç yapısına benzer. Model dahilindeki herhangi bir düğüm, altındaki n sayıda düğüme bağlanırken, kendisinin üstünde ancak bir düğüme bağlanabilir. Hiyerarşik yapının en tepesindeki düğüm noktasına kök denir ve bu düğümün sadece bağımlı düğümleri bulunur. Bu veri yapısını gösteren grafiğe de hiyerarşik tanım ağacı denir. Hiyerarşik bir veri yapısı ve tanım ağacı aşağıda verildiği gibi bir veri modeli oluşturur. Aşağıdaki yapı bir şirketin yönetim kademesinin hiyerarşik yapısını içermektedir.

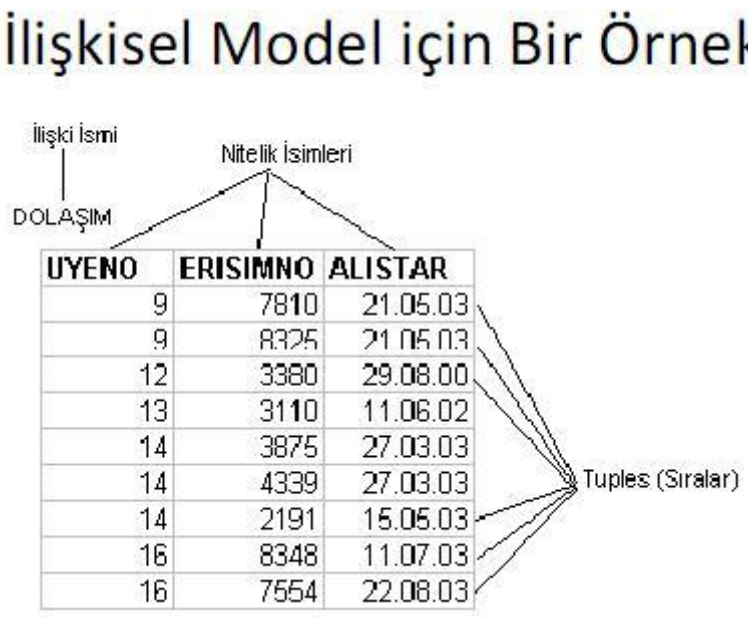


Hiyerarşik Tanım Ağacı

Hiyerarşik modelin PC ortamına uyarlanmış bir veritabanı yönetim sistemi şekli bulunmamaktadır. Bu kısımda ilişkisel veri modelini esas alan ilişkisel veritabanları önemli bir yer tutmaktadır.

İlişkisel Veri Tabanları:

1970'lerin başında E.F.Codd tarafından geliştirilmiş bir veri modeli şeklini esas alır. Bu sistemde veriler tablolar şeklinde saklanır. Bu veri tabanı yönetim sisteminde;veri alışverişi için özel işlemler kullanılır. Bu işlemlerde tablolar operandlar olarak kullanılır. Tablolar arasındaki matematiksel bağlantılarla(ilişkilerle) temsil edilen ilişkiler belirtilir. Günümüzde hemen hemen tüm veri tabanı yönetim sistemleri ilişkisel veri modelini kullanırlar. Bu model, matematikteki ilişki teorisine("the relational theory") dayanır. İlişkisel veri modelinde(relational data model) veriler basit tablolar halinde tutulur. Tablolar, satır ve sütunlardan oluşur. Sütunlar bilgi alanlarını, satırlar ise bilgilerin içeriğini belirler.



Neden veri tabanları kullanılır:

Veri Tabanlarının Avantajları:

Bilgisayar ortamında verilerin tutulması, saklanması ve erişilmesinde günümüze kadar değişik yöntem ve yaklaşımlar kullanılmıştır. Bu yaklaşımlardan biri olan Geleneksel Yaklaşım verilerin ayrı ayrı dosyalarda gruplanma yaklaşımını kullanmaktadır. Veri tabanı programlarından önce verileri saklamak için programlama dillerinde sıralı(sequential)ve rastgele(random) dosyalama sistemi kullanılırdı. Bu sistem; birbiriyle ilgili olan ve aynı gruba

dahil olan verilerin bir dosyada, bir başka gruba dahil olan verilerinde bir başka dosyada tutulması yöntemine dayanmaktadır. Verilerin artması, verilere aynı anda erişilmesi ve aynı anda erişilen verilerin erişenlere göre düzenlenmesi gibi ihtiyaçlar arttıkça geleneksel yaklaşım yetersiz kalmıştır. Geleneksek Yaklaşımın aşağıda listelenmiş bu ve buna benzer bir çok yetersizlik ve beraberinde getirdiği sorunlara alternatif olarak Veri Tabanı Yaklaşımı zamanla Geleneksel Yaklaşımın yerini almıştır. Günümüzde veriler Veri Tabanı Yaklaşımı ilkesine göre tutulmakta ve işlenmektedir.

Geleneksel Yaklaşımın(Dosya-İşletim sistemi) Sakıncaları

- Veri tekrarı ve veri tutarsızlığı
- Verinin paylaşılamaması
- Uygulamalardaki her yeni gereksinim ve değişikliğin yalnız uzman kişiler tarafından karşılanabilmesi
- Veriye erişim ve istenen veriyi elde etme güçlükleri
- Karmaşık veri saklama yapıları ve erişim yöntemlerini bilme zorunluluğu
- Bütünlük(integrity) sorunları
- Güvenlik, gizlilik sorunları
- Tasarım farklılıkları ve standart eksiklikleri
- Yedekleme, yeniden başlatma, onarma gibi işletim sorunları.

Veri Tabanı Yaklaşımının Yararları

- Ortak verilerin tekrarının önlenmesi; verilerin merkezi denetiminin ve tutarlılığının sağlanması
- Veri paylaşımının sağlanması
- Fiziksel yapı ve erişim yöntemleri yaklaşımlarının, çok katmanlı mimarilerle kullanıcılardan gizlenmesi
- Her kullanıcıya yalnız ilgilendiği verilerin, alışık olduğu kolay, anlaşılır yapılarda sunulması
- Sunulan çözümleme, tasarım ve geliştirme araçları ile uygulama yazılımı geliştirmenin kolaylaşması
- Veri bütünlüğü için gerekli olanakların sağlanması, mekanizmaların kurulması
- Güvenlik ve gizliliğin istenilen düzeyde sağlanması
- Yedekleme, yeniden başlatma, onarma gibi işletim sorunlarına çözüm getirilmesi.

- Veriler tek bir merkezde tutulur ve aynı veri her kullanılan değişik bilgisayarlarda tekrar tekrar tutulmaz. Programcı, kullandığı verilerin yapısı, organizasyonu ve yönetimi ile ilgilenmeden veri tabanının bunları kendinin koordine etmesi ve yönetmesidir. Veri bağımsızlığı, veri tabanı yönetim sistemi programlarının en temel amaç ve özelliklerindendir.

Bilinen VTYS programları:

MS SQL Server: Microsoft firması tarafından geliştirilen, bir orta ve büyük ölçekli VTYS'dir. ANSI SQL'e eklentiler yazmak için T-SQL'i destekler.

Oracle: Daha çok yüksek ölçekli uygulamalarda tercih edilen bir VTYS'dir. ANSI SQL'e eklentiler yapmak için PL-SQL adlı dil geliştirilmiştir.

Sybase: Bir orta ve büyük ölçekli VTYS'dir. ANSI SQL'e eklentiler yazmak için T-SQL komutlarını destekler. Ülkemizde daha çok bankacılık ve kamusal alanlarda tercih edilmektedir.

Informix: Bir orta ve büyük ölçekli VTYS'dir.

MySQL: Genellikle unix-linux temelli web uygulamalarında tercih edilen bir VTYS'dir. Açık kod (open source) bir yazılımdır. Küçük-orta ölçeklidir.

Postrage SQL: Bu da mysql gibi açık kod bir VTYS'dir.

MS Access: Çoklu kullanıcı desteği yoktur. İşletim sisteminin sağladığı güvenlik seçeneklerini kullanır. Bunun yanında belli sayıda kayda kadar.(1 milyon civarı) yada belli bir boyutun (yaklaşık 25 MB) altına kadar bir sorun çıkarmadan kullanılabilecek bir küçük ölçekli VTYS'dir.

VERİ MODELİ

Bir veri modeli, verinin hangi kurallara göre yapılandırıldığını belirler.

Varlık İlişki Veri Modeli(Entity-Relationship model):

Vİ yada ER modeli olarak isimlendirilen bu model 1976 yılında geliştirilmiştir.

Bugüne kadar varlık-ilişki modeline dayalı bir VTYS geliştirilmemiştir. Buna rağmen burada varlık ilişki modelinin verilmesinin temel sebebi, VTYS'den bağımsız veri çözümlemesinde ve semantik veri modellemede en çok kullanılan model olmasıdır.

Bu model kullanılarak önce;

- VTYS'den bağımsız olarak veriler çözümlenir,
- Veri modellemesi yapılır,
- Veriler ve veriler arasındaki ilişkilerin anlamları ve özellikleri incelenerek E-R çizelgeleri oluşturulur,
- Kullanılacak VTYS belirlenir ve daha sonra E-R çizelgeleri bu sistemin veri modellerine dönüştürülerek veri tabanı şemaları oluşturulur.

Varlık ve Varlık Kümesi

Varlık:

Var olan ve benzerlerinden ayırt edilebilen her nesneye **varlık (entity)** adı verilir. Varlık bağımsızdır ve tek başına tanımlanabilir. Bir veri tabanı uygulaması hakkında tanımlayıcı bilgi saklanabilen her şey varlık olarak kabul edilir. Bir varlık, ev, araba gibi bir nesne yada futbol maçı, tatil, satış gibi olaylar vb olabilir.

Örnek: Bir öğrenci, bir kitap, Veri Tabanı Yönetim Sistemleri Dersi, Burak birer varlıktır.

Varlık Kümesi:

Aynı türden benzer varlıkların oluşturduğu kümeye ise **varlık kümesi (entity set)** adı verilmektedir. Varlık kümeleri iç içe, kesişen yada ayrık kümeler olabilir.

Örnek: Öğrenciler, kız öğrenciler, Bilgisayar Programlama Bölümü Öğrencileri, yurtta kalan öğrenciler, renkler, dersler, yıllar, tarihler, satış miktarlar vb… varlık kümesi örnekleri olarak sayılabilir.

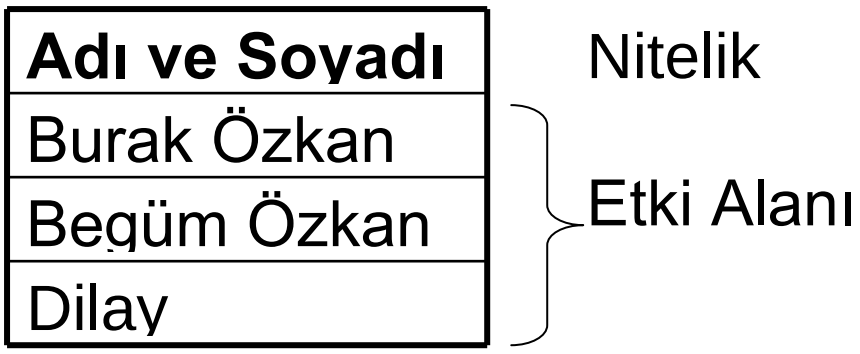
Nitelik:

Bir varlık kümesinde varlıkların özelliklerini göstermek ve varlıkları birbirinden ayırt etmek için **nitelikler(attributes)** kullanılır. Gerçek dünyada varlıkların çok sayıda özellikleri olabilir. Ancak veri modellemede, bu özelliklerin uygulamalarda gerekli olan küçük bir kısmı alınarak soyut bir modelleme yapılır.(Sicil no, adı, soyadı, adres vb..)

Etki Alanı:

Her niteliğin bir etki alanı (domain) vardır.Etki alanı ilgili niteliğin olabilecek tüm değerlerini içeren bir kümedir.

Örnek:



Türetilen Nitelik:

Bir nitelik kullanılarak bir başka varlık niteliği elde edilebiliyorsa, bu yeni niteliğe **türetilen nitelik** adı verilir.

Örneğin, "personel" varlığının "doğum tarihi" niteliğinden yararlanılarak "yaş" niteliği elde edilebilir. Bu örnekte "yaş" niteliği türetilen niteliktir, tasarımda ayrıca tanımlanmasına gerek yoktur.

Birleşik Nitelik:

Birden fazla nitelik birleştirilerek yeni bir nitelik oluşturulabilir. Bu tür niteliklere birleşik nitelik adı verilir.

Örneğin, "mahalle", "cadde", "sokak", "apartman", "posta kodu" ve "şehir" gibi nitelikler birleştirilerek "adres" isimli yeni bir nitelik oluşturulabilir.

İlişki:

Varlıklar arasındaki bağıntıya **ilişki (relationship)** adı verilir.

- İkili ilişki:
  - bir öğrenci ile bir ders (öğrenci dersi **alır**)
  - bir firma ile bir malzeme (firma malzemeyi **üretir**)
- Üçlü ilişki:
  - Bir işçi, bir ürün ve bir makine (işçi bu ürünü üretirken bu makineyi kullandığı için)

İlişki Kümesi:

Aynı türden ilişkilerin oluşturduğu kümeye ilişki kümesi adı verilir.

- Matematiksel olarak E1, E2, E3, ..., En varlık kümeleri arasındaki bir R ilişkisi aşağıdaki gibi tanımlanır:

$R = \{(e1, e2, e3, ..., en) : e1 \in E1, e2 \in E2, e3 \in E3, ..., en \in En\}$

Örnek:

Aşağıdaki iki varlık kümesini göz önüne alalım:

E1 = { Ali, Ayşe }

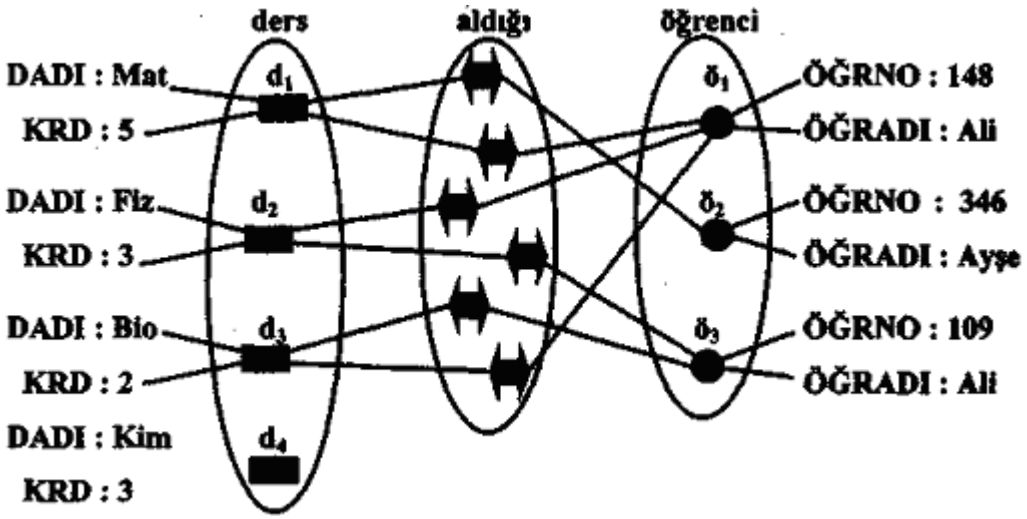
E2 = { Matematik, Fizik }

Bu varlık kümeleri için öğrenci ve aldığı ders ilişkileri aşağıdaki gibi ifade edilebilir:

R1 = { (Ali, Matematik) } R2 = { (Ali, Fizik) }

R3 = { (Ayşe, Matematik) } R4= { (Ayşe, Fizik) }

Aşağıdaki şekilde "öğrenci" ve "ders" varlık kümeleri arasında "aldığı" ilişkisi ile bir ilişki örneği görülmektedir:



Rol:

Aralarında ilişki kurulan varlıklardan her birinin ilişkideki işlevine varlığın rolü denir.

Farklı varlık kümeleri arasındaki ilişkilerde roller dolaylı yoldan anlaşılabildiği için çoğunlukla açıkça belirtilmez

- Örneğin, öğrenci ve ders arasında kurulan "aldığı" ilişkisinde varlıkların rolleri bellidir: öğrenci dersi alan, ders ise öğrenci tarafından alınandır.

Örnek:

İlişkilerdeki roller belirlenirken, başka niteliklere de bakmak gerekebilir.

- Örneğin "Personel" varlığında "ast-üst" ilişkisini belirlemek için "Görevi" niteliği dışında "Bölümü" niteliğine de bakmak gerekebilir.

Yönetici (üst, ast) ilişkileri:
(Begüm, Burak)
(Begüm, Dilay)
(Selin, Sezin)

Personel varlığı:

Adı	Bölümü	Görevi
Burak	Muhasebe	İsci
Begüm	Muhasebe	Yönetici
Dilay	Muhasebe	İsci
Selin	Satış	Yönetici
Sezin	Satış	İsci

- **Yapılar:**

Soyutlama, küme ve ilişki veri yapılarının temel unsurlarıdır. Detayları gizleme ve genel üzerinde yoğunlaşma yeteneği olan soyutlama veriyi yapılandırma ve görüntüleme işlemini yapar ve veri kategorilerini elde etmek için kullanılır. Yapılar düzgün şekilde tanımlanmış veri gruplarıdır. Kendisi de aynı zamanda bir küme olan ilişki, iki nesne arasındaki ilişkiyi gösteren bir tip olarak kümelerin toplanmasını ifade eder. Örneğin, ÖĞRENCİ ve OKUL arasında bir NOT ilişkisi vardır.

Veri yapısı oluşturulurken, verideki nesneler ve onlar arasındaki ilişkiler tablo ile temsil edilir. Veri tabanında uygulanabilecek genel kayıt ilişkilendirme tipleri şu şekilde sıralanabilir:

Bire bir ilişkiler (One-to-one relationships): Aralarında ilişki olan iki tablo arasında, tablolardan birindeki asıl anahtar alanın kayıt değerinin, diğer tablodaki sadece bir kayıtta karşılığının olması durumunu gösteren ilişki tipidir.

Örnek: Bir öğrencinin doğum yeri bilgisinin doğum yerleri tablosunda bir şehre karşılık gelmesi gibi.

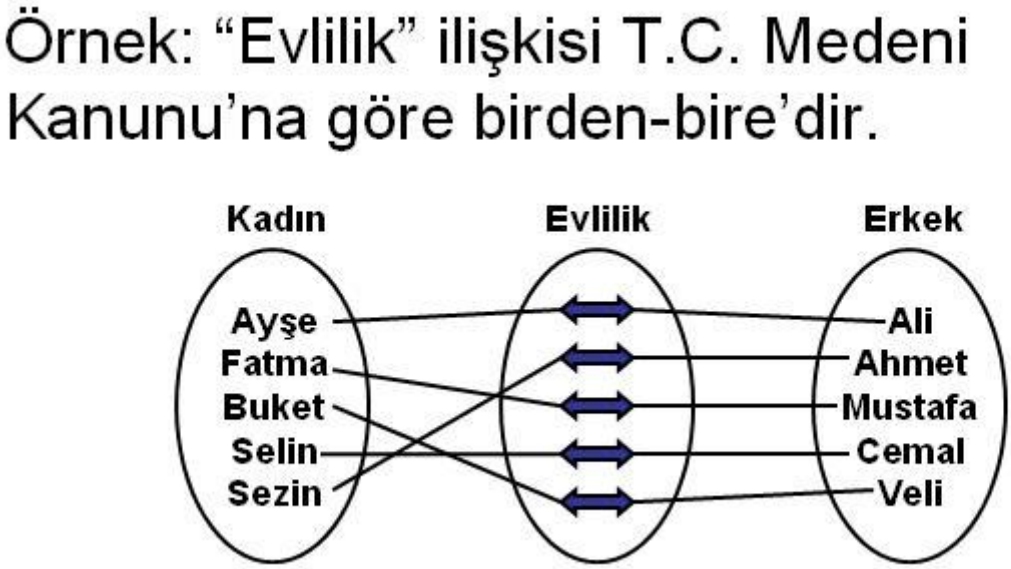
Aşağıdaki şekilde bir bire bir ilişki örneği görülmektedir.



1 Çalışanlar Tablosunda her Futbolcuyla eşleşen BİR kayıt vardır.

2 Bu değer kümesi, ÇalışanNo alanının ve Çalışanlar Tablosunun bir alt kümesidir.

Bir başka bire bir ilişki örneği evlilik ilişkisi üzerinden verilebilir. Aşağıdaki şekilde kümeler ve ilişki yönü ile birebir bir ilişki olarak gösterilmiştir.



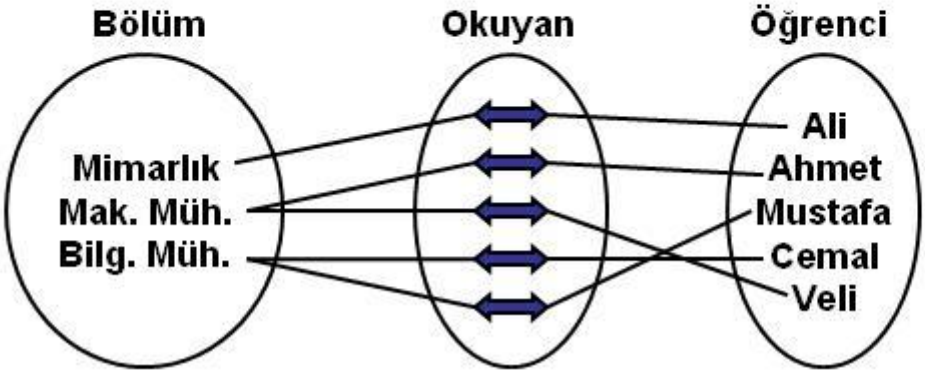
Tekil çoklu ilişkiler (One-to-many relationships): Aralarında bir ilişki bulunan iki tablo arasında, anahtar alanının kayıt değerinin, diğer tablodaki birden fazla kayıtta karşılığının olması durumunu gösteren ilişki tipidir.

Örnek: Bir öğrencinin birden fazla almış olduğu derse ve bu derse ait vize, final ve sınav sonuçları gibi. Bir öğrenciye karşılık birden fazla ders notu. Aşağıda birden çoğa bir ilişki türü olarak kitaplar ile yazar arasındaki ilişki örneği verilmiştir. Bir yazar birden çok kitap yazmış olabilir. Ancak her kitap mutlaka bir yazar tarafından yazılmıştır.



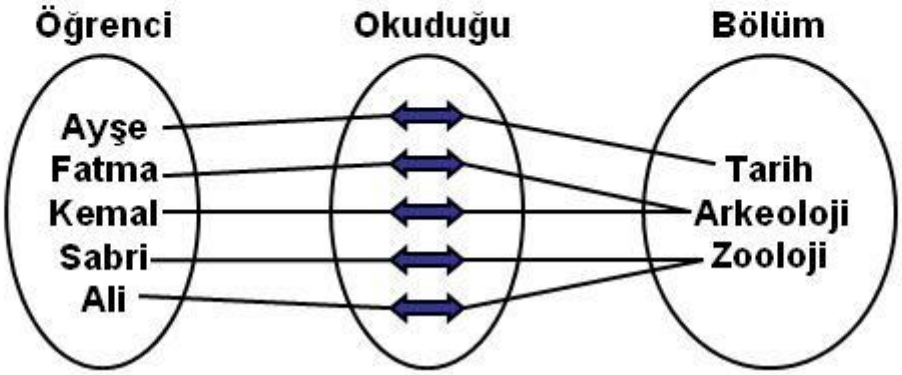
Birden çoğa bir başka ilişki örneği aşağıdaki şekilde gösterilmiştir. Bölümle öğrenci arasında bulunan ”okuyan” ilişkisi örneği:

Ornek: “Bölüm” ve “Oğrenci” varlık kümeleri arasındaki “Okuyan” ilişkisi, bölümden öğrenciye doğru birden-çoğa şeklindedir.



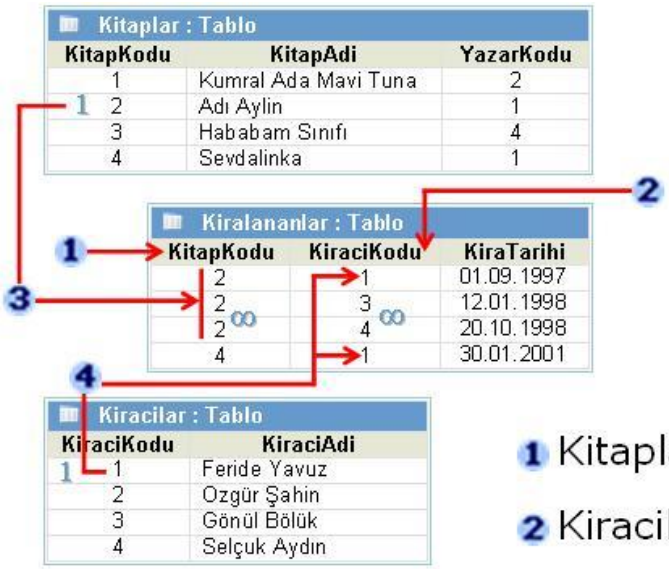
Çoğul tekli ilişkiler (Many-to-one relationships): Aralarında bir ilişki olan iki tablo arasında, tablolardan birindeki bir kaydın değerinin, asıl anahtar alanının olduğu diğer tabloda, birden fazla kayıtta karşılığının bulunması durumunu gösteren ilişki tipidir. Çoktan bire bir ilişki şekli aşağıda verilmiştir. Bölümle öğrenci arasında “okuduğu” ilişkisi ile belirtilen bir ilişki şu şekilde olmaktadır:

Örnek: “Öğrenci” ve “Bölüm” varlık kümeleri arasındaki “Okuduğu” ilişkisi, öğrenciden bölüme doğru çoktan-bire şeklindedir.



Çoklu ilişkiler(Many-to-many relationships): Aralarında bir ilişki bulunan iki tablo arasında, tablolardan herhangi birindeki herhangi bir kaydın, diğer tablodaki birden fazla kayıtla ilişkilendirilebildiği ilişki tipidir.

Aşağıdaki şekilde çoklu ilişkilerin örneği görülmektedir.



1 Kitaplar Tablosundaki Birincil Anahtar

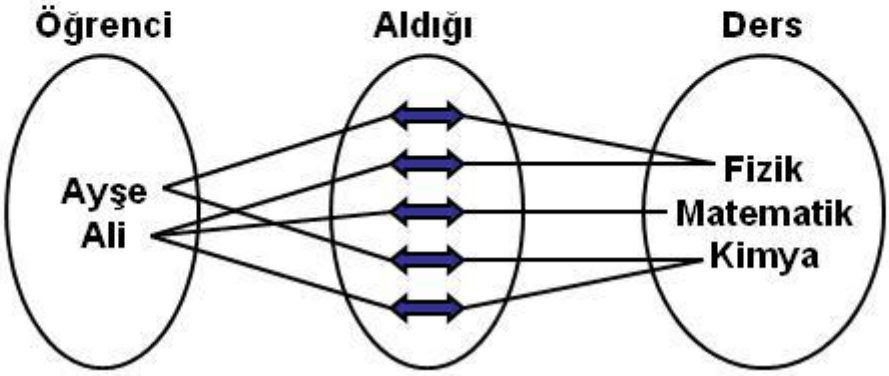
2 Kiracilar Tablosundaki Birincil Anahtar

3 Bir Kitap Bir-Çok kiralamada yer alabilir ...

4 ... her Kiracı Bir-Çok Kiralamada görülebilir.

Aşağıdaki resimde bir çoktan çoğa ilişki örneği öğrenci ve ders varlıkları arasında verilmiştir. Birden fazla öğrenci birden fazla ders alabilmektedir.

Örnek: "Öğrenci" ve "Ders" varlık kümeleri arasındaki "Aldığı" ilişkisi, çoktan-çoğa şeklinde bir ilişkidir.



- **Kısıtlar:**

Veriler üzerindeki sınırlamalara kısıt(kısıtlama) adı verilir. Kısıtlar, veri modellerinde bütünlük sağlamak için kullanılırlar. Örneğin "öğrenciye ait not bilgisinin 0-100 arasında olması " gibi.

- **İşlemler:**

İşlemler bir veri tabanı durumundan bir başka veri tabanı durumuna geçmek için kullanılan işlemlerdir. Bunlar verinin çağrılması, güncellenmesi, eklenmesi veya silinmesi ile ilgili işlemlerdir. Bütünlük mekanizması, toplam fonksiyonları (istatistiksel fonksiyonlarda bunlar arasındadır), veriye ulaşım kontrolleri gibi genel işlemler vardır. CODASYL tarafından yayınlanan bu mekanizmalara veri tabanı yöntemleri denir.

Ödev: 1-CODASYL nedir araştırınız.

2-Verilen ilişki türlerine(birden bire, birden çoğa, çoktan bire ve çoktan çoğa ilişkileri) **en az** birer örnek geliştiriniz. Örneklerinizi varlık kümelerini ve ilişki adını belirterek, derste verildiği gibi şekille gösteriniz.

3- Veri tabanı kısıtlamalarına yeni örnekler üretiniz. Hangi alanları nasıl bir kısıtlamaya tabi tutabilirsiniz. Örnekler üzerinden açıklayınız.

Anahtarlar:

Bir varlık kümesi içindeki varlıkları ya da bir ilişki kümesi içindeki ilişkileri birbirinden ayırt etmek için kullanılan nitelik ya da nitelik grubuna bu varlık ya da ilişki kümesinin anahtarı denir.

Anahtar, hem varlık kümeleri hem de ilişki kümeleri için geçerli bir kavram olsa da, daha çok varlık kümeleri için kullanılır.

Anahtar Türleri:

- **Süper anahtar (superkey)**
  - Değerleri ile bir kümedeki varlıkları (veya ilişkileri) ayırt etmeyi sağlayan niteliğe (veya nitelik grubuna) bu varlık / ilişki kümesinin süper anahtarı denir. Ayırt etme özelliğine sahip olmak için gereğinden fazla nitelik içerebilir.
- **Aday anahtar (candidate key)**
  - Eğer bir varlık / ilişki kümesinin süper anahtarının bir altkümesi de bu varlık / ilişki kümesini ayırt edebiliyorsa, bu altküme aday anahtardır (ya da kısaca anahtardır).

Örnek:

Eğer bir üniversitede tüm öğrencilerin numaraları birbirinden farklı ise, öğrencileri ayırt etmek için öğrenci numarası yeterlidir.

Bu durumda "öğrenci numarası", "öğrenci" varlık kümesi için aday anahtar ya da kısaca anahtardır, içinde öğrenci numarası bulunan her nitelik grubu (örneğin "öğrenci numarası", "adı" ve "soyadı") ise bu varlık kümesinin süper anahtarıdır.

Güçlü ve Zayıf Varlık Kümeleri:

Her varlık kümesi için bir anahtar bulmak mümkün olmayabilir.

Eğer bir varlık kümesinin niteliklerinden en az bir anahtar oluşturulabiliyorsa, bu varlık kümesine **güçlü (strong**) varlık kümesi denir.

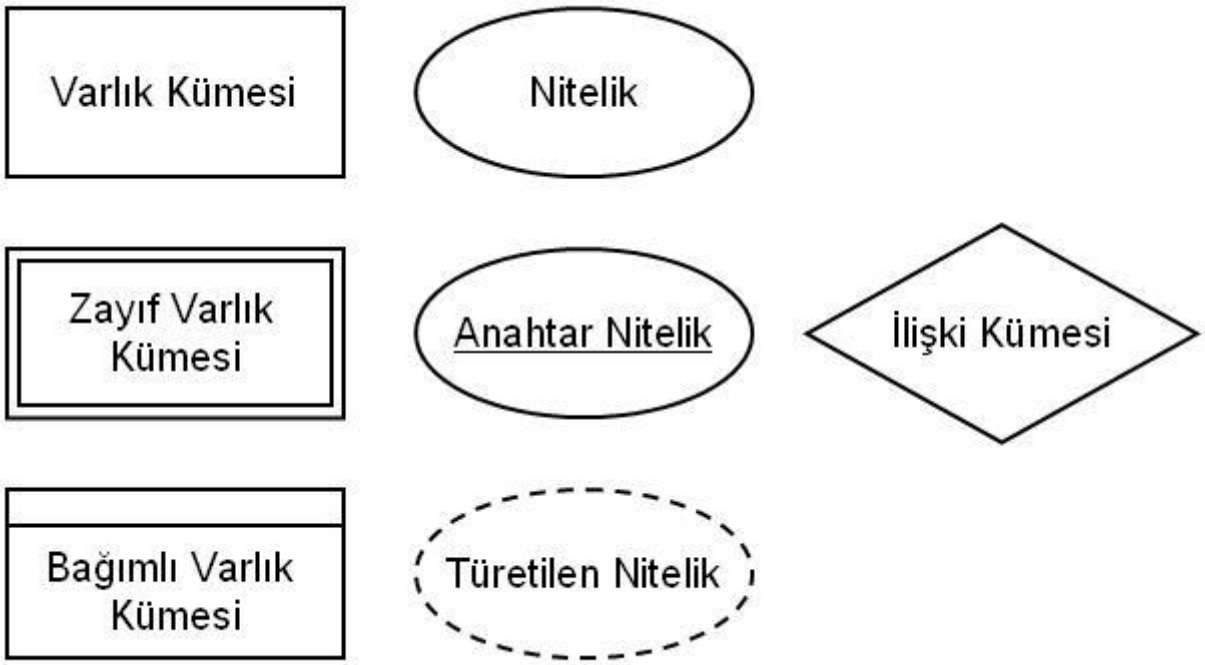
Eğer bir varlık kümesinin niteliklerinin tümü alınsa bile bir anahtar oluşturmuyorsa bu varlık kümesine **zayıf (weak)** varlık kümesi denir

Zayıf Varlığı Güçlendirme:

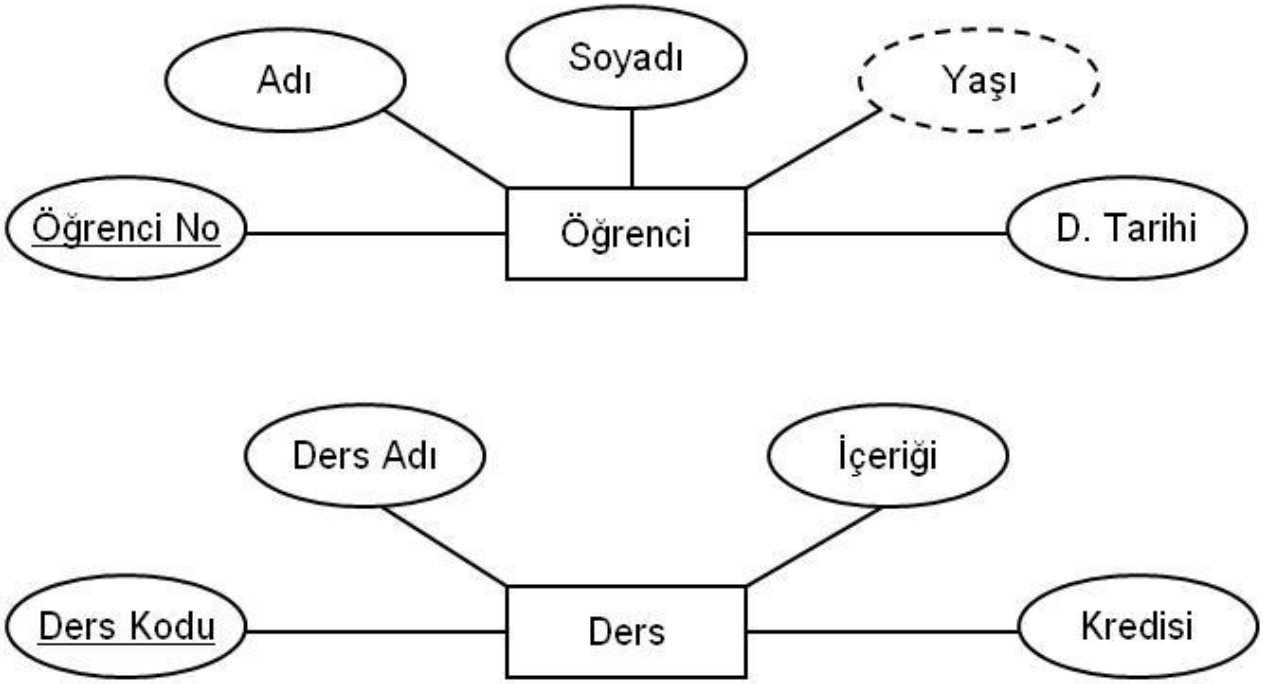
Türkiye'deki tüm lise öğrencilerinin bilgilerini içeren ÖĞRENCİ varlık kümesi zayıf bir varlık kümesidir. Çünkü farklı liselerde öğrenci numarası, adı ve soyadı aynı olan öğrenciler bulunabilir.

ÖĞRENCİ varlık kümesi ile LİSE varlık kümesi arasında bir OKUYAN ilişkisi kurulursa, öğrencileri birbirinden ayırdetmek için kullanılan ÖĞR_NO niteliğine, LİSE varlık kümesinin anahtarı olan LİSE_KODU eklenir. Bu durumda ÖĞRENCİ varlık kümesinin anahtarı (LİSE_KODU, ÖĞR_NO) ikilisi olur.

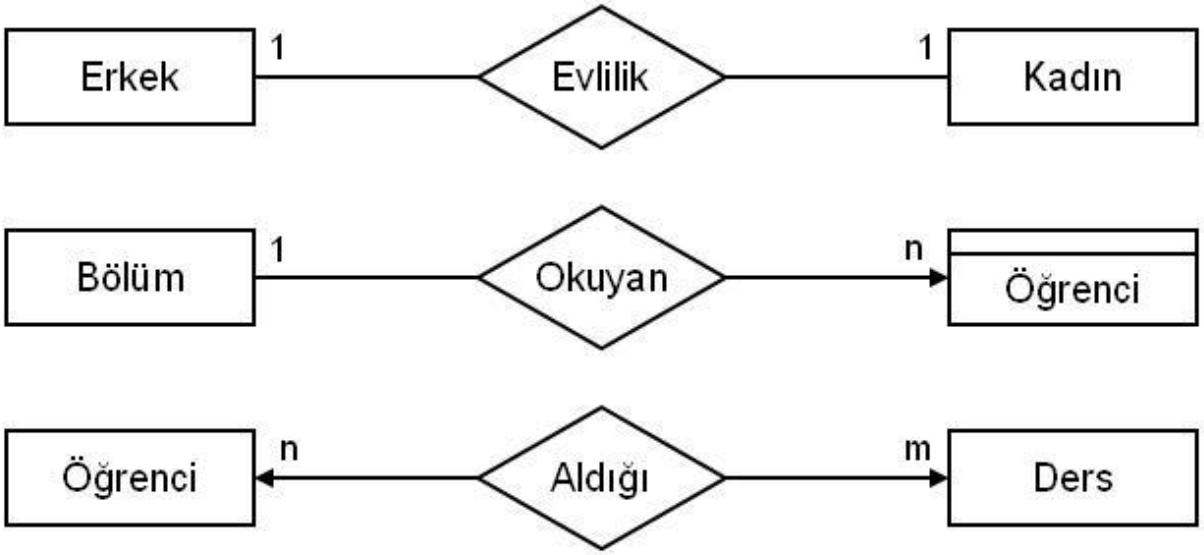
Varlık İlişki Çizelgeleri:



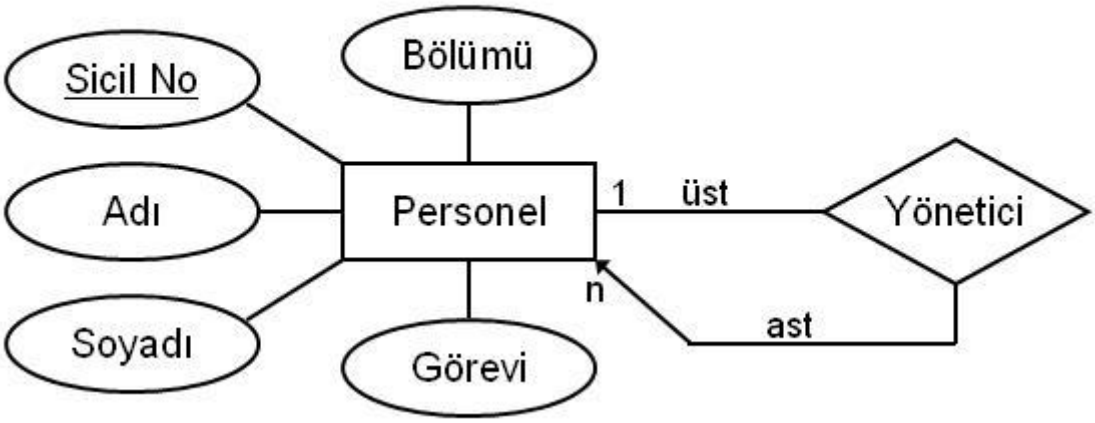
Varlık Kümesi ve Nitelik Örnekleri:



İlişki Kümesi Örnekleri:

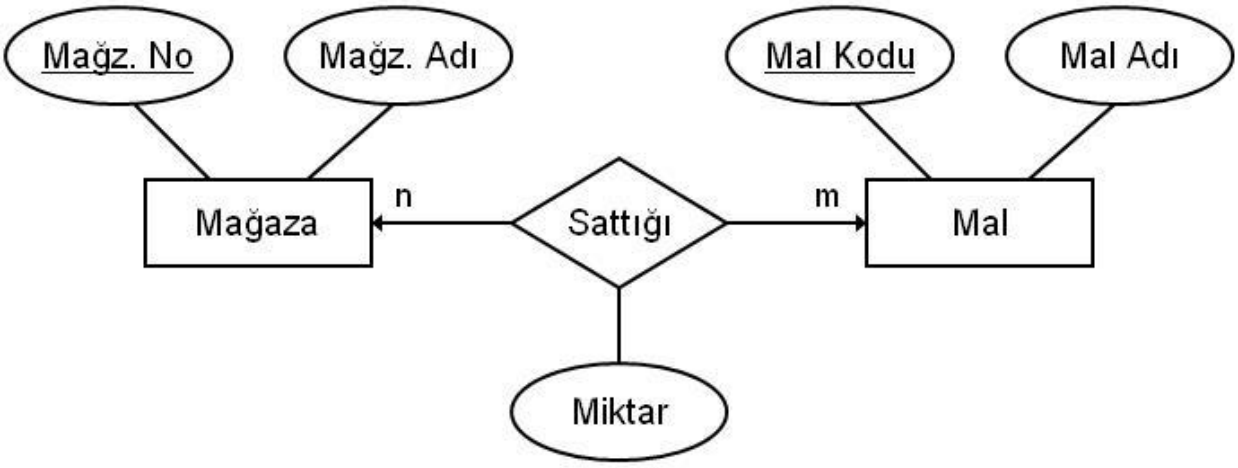


Rol Örnekleri:



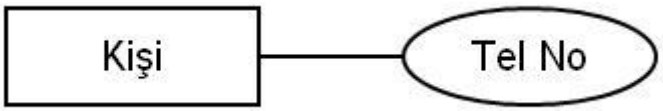
İlişkilerde Nitelik:

- İlişkilerde de tanımlayıcı nitelikler bulunabilir.
- Aşağıdaki şekilde "Miktar" niteliği "Sattığı" ilişkisi için tanımlayıcı niteliktir.



Örnek Veri Modelleri:
Örnek: Telefon Numarası:

- "telefon numarası" **kişi** varlık kümesinin bir niteliği olarak düşünülürse;
  - "telefon numarası" kişilerden bağımsız olarak var olamaz.
  - bir kişinin sadece bir telefon numarası bulunabilir.
  - birden çok kişinin telefon numarası aynı olabilir (telefon numarası kişi varlık kümesinin anahtarlarından biri olarak tanımlanmadığı sürece).



- "telefon numarası" ayrı bir varlık kümesi olarak düşünülüp, bu varlık kümesi ile kişi varlık kümesi arasında ilişki kurulursa;
  - telefonun numarası dışında nitelikleri de bulunabilir.
  - kişi ve telefon varlık kümeleri arasındaki ilişkinin türüne göre her kişinin bir ya da birçok telefonu olabilir.
  - bir telefon numarası bir ya da birçok kişiye verilebilir.

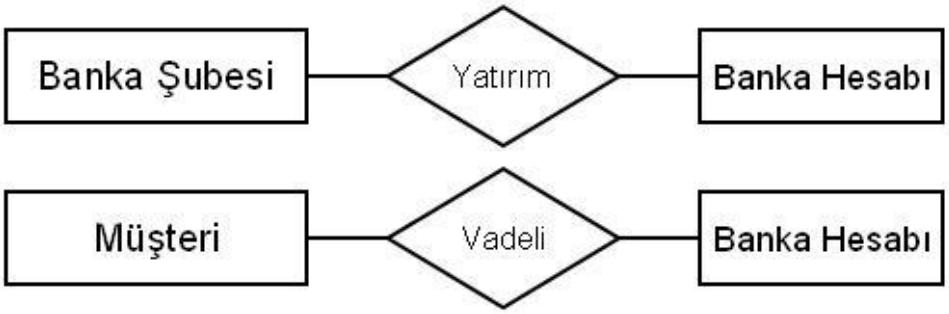


Örnek : Banka Hesabı

- "Banka hesabı" banka şubesi ve müşteri varlık kümeleri arasında bir ilişki olarak düşünülebilir.



- "Banka hesabı" ayrı bir varlık kümesi olarak düşünülüp bu varlık kümesi ile banka şubesi ve müşteri varlık kümeleri arasında birer ilişki de kurulabilir.

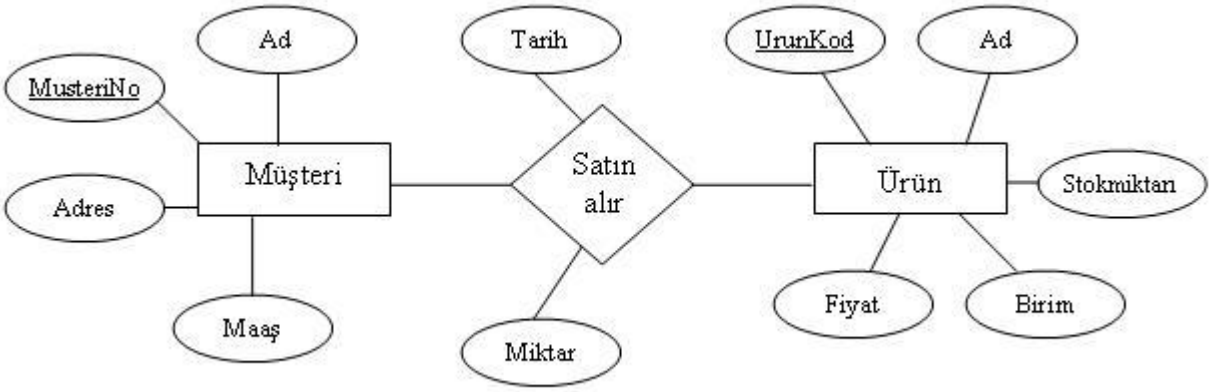


Veri Modellemede Seçim:

- Veri modellemede varlık kümelerinin, niteliklerin ve ilişki kümelerinin seçimi çok önemlidir, Ancak bunların nasıl seçileceğine ilişkin kesin kurallar da yoktur.
- Kuruluşun öncelikleri ve uygulamaların özellikleri yanında veri modellemeyi gerçekleştiren bilişim teknik personelinin anlayışı da düzenlemede etkili olmaktadır.

Ödev:

4-Aşağıdaki diyagramı derste anlatılanlara göre detaylı bir şekilde açıklayınız.



5- Varlık ilişki diyagramlarından tablolamaya geçiş hakkında bilgi edininiz.

İlişkisel Veri Modelleri:

İlişkiler ve onların temsilleri olan tablolardan oluşan ilişkisel veri modeli ilk olarak 1970 yılında Codd tarafından ortaya atılmıştır. İlişkisel veri modellerinde kullanılan tek yapılandırma aracı ilişkidir. Aşağıdaki örneklerde büyük harflerle yazılan ifadeler ilişki isimlerini, parantez içindeki ifadeler de tanım kümesini göstermektedir.

UNIVERSITE (UniversiteKodu, UniversiteAdi, Adres)
OGRETIMELEMANI(OgrKodu, OgrAdi, OgrSoyadi, Bolum, Brans)
OGRENCI(OgrenciNo,OgrenciAdi, OgrenciSoyadi,Bolum, Adres)
DERSLER(DersKodu, DersinAdi, Kredisi, Yariyil)
OGRENCINOTLAR(OgrenciNo, DersKodu,Vize, Final)

Yukarıdaki satırlar, basit bir üniversite veri tabanının ilişkisel şemasını göstermektedir. İlişkisel şema, ilişki isimlerinin ve karşılık gelen tanım kümesi isimlerinin listesidir. Varlık tiplerini belirlemekte kullanılır.

Universite

UniversiteKodu	UniversiteAdi	Adres
2300	Fırat Üniversitesi	Elazığ
0600	ODTU	Ankara
3400	Fatih Üniversitesi	İstanbul

Şekil:İlişkisel Tablo

Veri Tipleri(Data Types)

MS Access İçin Veri Tipleri:

Veri Türü	Anlamı
Text (Metin)	Alfabetik yada alfasayısal verileri tanımlar. En fazla 255 karakterlik veriler içerebilir. Bu veriler üzerinde matematiksel işlemler yapılamaz.
Memo (Büyük metin)	Tablo içindeki uzun açıklamalar veya notların tutulması için kullanılır.Bu alanlara 32000 karakter uzunluğundaki metinler kaydedilebilir.
Number (Sayı)	Sayısal alanların tanımlanması amacıyla kullanılır. Bu tür verileri içeren alanlar üzerinde matematiksel işlemler yapılabilir.
Date/Time (Tarih/Saat)	Tarih veya saat bilgilerini içeren alanların tanımlanması amacıyla kullanılır.Bu alanlar baytlık yer işgal ederler.
Currency	Özellikle büyük rakamsal değerlerin kullanıldığı alanlardır.
AutoNumber (Otomatik Sayı)	Bu veri türüne sahip alanlar, tabloya yeni bir kayıt eklendiğinde Access tarafından otomatik olarak üretilen sıralı yada raslantısal bir sayısal değer içerirler. Dolayısıyla bu alanda belirtilen sayısal değerler tektir ve ayrı kayıtlarda birbirinin aynısı olamaz. Bu tür alanlar kullanıcı tarafından güncelleştirilemez.
Yes/No	Bir baytlık bir uzunluğa sahip olan bu alanlar Yes/No biçimindeki verilerin saklanması amacıyla kullanılır.
Ole Object	Eğer tablonun bir alanında resim, ses veya grafik gibi OLE nesnelerinin saklanması söz konusu ise böyle bir tanım yapılır. Bu alanın büyüklüğü 1 GB kadar olabilir.
Hyperlink	Bu veri türü ile herhangi bir Web sitesini bu alandaki veri olarak tanımlayabiliriz. WWW'de bulunan bilgilerle direkt bağlantı sağlanabilir.
Lookup	Bir başka tablo ile bağlantı kurularak açılan listeden bilgi

Wizard (AramaSihirbazı)	seçilmesini ve tanımlanan alana taşınmasmını sağlar.

MYSQL İçin Veri Tipleri:

TEXT - YAZI
CHAR( ) : Sabit 0 - 255 karakter.
VARCHAR( ) : Değişken 0 - 255 karakter.
TINYTEXT : En Fazla 255 karakter.
TEXT : En Fazla 65.535 karakter.
BLOB : En Fazla 65.535 karakter.
MEDIUMTEXT : En Fazla 16.777.215 karakter.
MEDIUMBLOB : En Fazla 16.777.215 karakter.
LONGTEXT : En Fazla 4.294.967.295 karakter.
LONGBLOB : En Fazla 4.294.967.295 karakter.

SAYILAR
TINYINT( ) : -128 ,127 yada 0-255 UNSIGNED.
SMALLINT( ) : -32.768 ,32.767 yada 0 – 65.535 UNSIGNED.
MEDIUMINT( ) : -8.388.608 , 8.388.607 yada 0 – 16.777.215 UNSIGNED.
INT( ) : -2.147.483.648 , 2.147.483.647 yada 0 – 4.294.967.295 UNSIGNED.
BIGINT( ) : -9.223.372.036.854.775.808 , 9.223.372.036.854.775.807 yada 0 – 18.446.744.073.709.551.615 UNSIGNED.
FLOAT : Küçük Noktalı sayı.
DOUBLE(,) : Büyük Noktalı sayı.
DECIMAL(,) : DOUBLE tipte string şeklinde saklanır.

TARİH , SAAT
DATE : YYYY-MM-DD
DATETIME : YYYY-MM-DD HH:MM:SS
TIMESTAMP : YYYYMMDDHHMMSS
TIME : HH:MM:SS

DİĞER
ENUM () : Kullanıcı tanımlı liste tipi. Ör; ENUM('e','h')
SET : Küme Tipi. ENUM benzeri. Aynı anda birden fazla kayıt tutabilir.

ORACLE VERİ TİPLERİ :

CHAR(karakterSayisi) : Maximum 255 karakterlik sabit uzunluktaki alfanümerik verilerin tutulabileceği alandır.
DATE : tarih ve saat tutan alandır. Ülke kodu desteği vardır. Standart olan veri tipi DD-MON-YY (31-JUL-05)dir.
MSLABEL : Trusted Oracle?da kullanılan işletim sistemine ait binary dosyadır.
NUMBER (toplam,ondalık) : Sayısal verilerin tutulduğu alanlar için kullanılır. İlk hane toplam karakter sayısını(ondalık dahil), ondalık bölümü ise ondalık kısmın uzunluğunu belirtir.
NUMBER(hane) : Ondalık içermeyen tam sayılar için kullanılan veri tipidir.
NUMBER : herhangi bir sayı girilmeden belirtilen sayısal alan tipidir. Tavsiye edilmemekle birlikte, oracle tarafından desteklenen maximum sayısal değere kadar veri girilebilir.
VARCHAR2 (sayı): Maximum 4000 karakterli değişken uzunluktaki alfanümerik dataların tutulabildiği alanlar için kullanılır.
LONG : 2 GB? a kadar karakter bilgi tutabilen bir alan türüdür. Bir tabloda birden fazla long veri tipine sahip alan olamaz. Bu alan üzerinde indeks oluşturulamaz. (*WHERE *GROUP BY * ORDER BY * DISTINC * CREATE CLUSTER *CREATE TABLE AS SELECT *SUBSTR, INSTR gibi SQL cümlelerinde kullanılamaz.)
LONG RAW : 2 GB? a kadar binary bilgi tutabilen alanlar için kullanılır.
RAW (sayı) : Maksimum 255 byte?a kadar bilgi tutabilen binary alanlar için kullnılır.
ROWID : Oracle´ın, tablodaki her bir satır için oluşturduğu sıra numarasıdır. Oracle tarafından otomatik oluşturulur.

MS SQL SERVER Veri Tipleri:

SQL Server 2005 için veri tipleri aşağıdaki gibidir.

1. **Exact Numeric**: Sayıları eğer varsa ondalık kısımlarıyla birlikte depolayan veri tipleri
2. **Approximate Numeric** : Exact Numeric tipinin tutamadığı uzunluktaki ondalık sayıları tutan veri tipleri
3. **Monetary** : Parasal birimleri tutmak için kullanılan veri tipleri
4. **Date and Time** : Tarihsel değerlerin tutulduğu veri tipleri
5. **Cahracter** : Uzunluğuna göre karakterleri depolayan veri tipleri (kelimeler… gibi)
6. **Binary** : Binary tipli verileri tutmak için kullanılan veri tipleri (resim, mp3…gibi)
7. **Special Purpose** : Özelleştirilmesi gereken karmaşık veri tiplerini tutmak için tasarlanmış veri tipleridirler.

1. Exact Numeric Data Tipleri

Bu veri tipleriyle tam sayılar veya ondalık sayılar tutlabilirler. Bu veri tipleriyle matematik işlemleride yapılabilir. Kapladıkları alana göre ve kapasitelerine gore 6 ya ayrılırlar:

Bigint = 8 bayt yer kaplar. (int: Integer(tamsayının kısaltması) Adından belli olduğu üzere tam sayıları tutar. Tutabileceği aralıkda -2^{63} ile $2^{63}-1$ arasıdır. (Pek kullanılmaz)

Int= 4 bayt yer kaplar. Yine belirli aralıktaki tam sayıları tutarlar. Aralıkda -2^{31} ile $2^{31}-1$ dır. En çok kullanılan veri tiplerinden biridir. Bizim için sadece bir tane olması gereken değerlere (ID lere) verilebilir.

Smallint=2 bayt yer kaplar. -2^{15} ile $2^{15} - 1$ arasındaki değerleri tutar. (-32768 ile 32767 arası)

Tinyint=1 bayt yer kaplar. 0 'dan 255 'e kadar olan sayıları tutarlar.

Decimal(p,s)=Decimal ondalık veya tam butun sayıları tutar ama ondalık sayı için kullanılır. Neden? Çünkü kapladığı alan bakımından. Decimal içine parametre alır. Yani şöyle ki mesela decimal(4,2) diye bie değişken tanımladım, bu demek oluyorki önce 4 basamak yazıyorum ondan sonra sağdan 2 basamak sayıp virgülü koyuyorum.

Örnek: decimal(5,3) önce rastgele beş basamak sayı yazıyorum "12345" , şimdide 3 basamak ayırıyorum "12,345". Yani decimal(5,3) sayısı 2 basamaklıdır ve virgülden sonra 3 basamak bulundurur.

Numeric(p,s) = İşlev ve özellik bakımından decimal ile aynıdır.(Pek kullanılmaz)

2. Aproximate Numeric Data Tipleri

Bu veri tiplerinde (zaten 2 tane var) amaç decimal in tutamadığı büyüklükteki değerleri tutmaktır.

Float(n) = -1,79E + 308 'den 1,79E + 308 'e kadar olan değerleri alabilir. Kapladığı alan içine aldığı "n" değerine göre değişir. (1<= n <= 53 olmalı)

Real = eski versiyonlarda kullanıldığı için bu versiyonda da kullanılmış. Bu da -3,40 + 38 'den 3,40 + 38 'ekadar olan küsürlü verileri tutar. 4 bayt yer tutar.

3. Monetary Data Tipleri

Söyleyecek fazla bişey sanırım, adları üstünde; para yerine kullanacağımız değişkenler

Money= -922 337 203 685 47,.5808 'den 922 337 203 685 477,5807 'ye kadar paraları tutabilir. 8 bayt yer kaplar.

Smallmoney= 214 748,3648 'den 214 748,3647 'kadar olan sayıları tutabilir. 4 bayt yer kaplar.

4. Date and Time Data tipleri

Tarih ve zamanı tutmak için kullanılan veri tipleri

Datetime= 1 Ocak 1753 'ten 31 Aralık 9999 'a kadar olan tarih aralığıı tutabilir. 8 bayt yer kaplar.

Smalldatetime= 1 Ocak 1900 'den 6 Haziran 2079 'a kadar olan tarih aralığını tutar. 4 bayt yer kaplar.

5. Character Data Tipleri

Karakter data tipleri en çok kullanılan data tipleridir. Kelimelerin yerine daha doğrusu harflerin yerine kullanılır. Parametre alırlar. Aldıkları parametreler içerdikleri karakter sayısıdır. İsimlerinde genelde "char " kelimesi vardır. Bunun başına "n" veya " var" önekleri getirilerek diğer data tipleri oluşturulmuştur. Kısaca bunlar ne demek bi bakalım.

Önce "n" varsa veri tipinin başında bu unicode karakterleride içeriyor demektir. Yani her dilde görüntülenebilir. Başında " n " olanlar olmayanlarına göre 2 kat daha fazla yer tutarlar (açıklayınca daha rahat görebilirsiniz).

Başında "var" olanlar ise içine aldıkları parametreye kadar genişletilebilirler. Örneğin varchar(5) direk olarak 5 baytlık yer ayırmaz, içine 3 harf yazarsak 3 bayt, 2 harf yazarsak 2 bayt yer ayırır. Bu neden boyle diyecek olursanız veri tipi varchar(5) değilde char(5) olsaydı ben içine tek bi karakter bile yazsam direk 5 baytlık alnı işgal edecekti. Şimdi diceksinizki o zaman ben hepsini varchar la yaparım. Ne kadar girersem o kadar yer kaplasın. Ama bu sizin yapacağınız sorgularda performans kaybetmenize sebep olabilir. Veri tiplerine geçelim..

Char(n) = Enfazla 8000 karakter alabilir. Dolayısıyla maksimum 8000 bayt yer kaplar. Bu veri tipi içine girilecek olan karakter sayısı kesin belli olduğu zaman kullanılır. Örnek: Northwind tablosundaki CustomerID ler (hepsi 5 karakterli)

Nchar(n) = Enfazla 4000 karakter alır. 2 ile 8000 bayt arası yer tutar. (unicode karakterleri yüzünden)

Varchar(n) = Enfazla 8000 karakter içerir. 8000 bayta kadar yer tutabilir. Bu veri tipini girceğimiz değerin uzunluğu kesin belli olmadığında kullanabiliriz. Örnek: şehirlerin isimleri…

Nvarchar(n) = Enfazla 4000 karakter alır. 2 ile 8000 bayt arası yer tutar. (unicode karakterleri yüzünden)

Varchar(max) = 2Gb 'a kadar karakter alabilir. 1 073 741 824 karakter alabilir.

Nvarchar(max) = 2Gb 'a kadar karakter alabilir. 536 870 912 karakter alabilir. (unicode karakterleri yüzünden)

Text = varchar(max) ın aynısı.

Ntext = nvarchar(max) ın aynısı. Bu text ve ntext çok fazla kullanılmıyor. Onların yerine varchar, char … kullanılıyor.

6. Binary Data Tipleri

Bu data tiplerini anlatmanın bence en klay yolu bunları depoya benzetmektir. İçine tutabileceği kadar ne koysanız tutar.Örneğin mp3, resim, word belgesi…gibi. Bunlarda parametre alıyor ve yukardaki "var" olayı bunlarda da var. (alanı direk ayırmayıp, biz değer girdikçe artan olay) Zaten 4 tane var bunlardan.

Binary(n) = 8000 bayta kadar veri tutabilir.

Varbinary(n) = 8000 bayta kadar veri tutabilir.

Varbinary(max) = 2Gb 'a kadar veri tutabilir.

Image = 2Gb 'a kadar veri tutabilir.(image=varbinary(max))

7. Specialized Data Tipleri

Hepsinin adını yazayım ama 3 tanesine değineceğim sadece. Bit, timestamp, uniqueidentifier, sql_variant, cursor, table, Xml.

Bit= 0 veya 1 tam sayı değeri alan değişkenlerdir. Yani geriye true veya false bi değer döndürür. Örneğin cinsiyette, evli-bekar, evet-hayır, var-yok gibi şekillerde kullanılabilir.

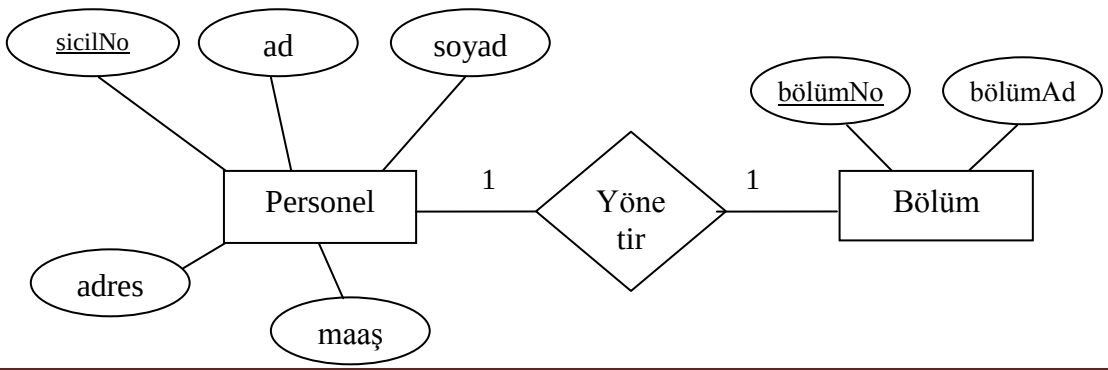
Sql_variant= Bu gireceğimiz değerin çeşidini bilmediğimiz sütunlarda kullanabileceğimiz bir veri tipidir. Yani resim, string, sayı, table ne olduğunu bilmediğim şeyler yerine kullanabilirim. Tabiki 8000 bayta kadar.

Uniqueidentifier= 16 bayt yer kaplar. Global tek değişkenlerdir (GUID, Globally Unique Identifier).

ER şemasının tablolara dönüştürülmesi

Bire-bir ilişkilerin dönüştürülmesi

Birebir ilişkiyi oluşturan varlık kümeleri tablolara dönüştürülür. Nitelikleri tabloların alanlarına dönüşür. Uygun olan varlık kümesinin anahtar alanı diğer varlık kümesine yabancı anahtar olarak eklenir. Bire-bir ilişkide belirtilen tanımlayıcı nitelikler, yabancı anahtar eklenen tabloya alan olarak eklenir. Örneğin bölüm ve personel varlık kümeleri arasındaki yönetir birebir ilişkisinde iki şekilde dönüştürme yapılır. Birincisi, her iki varlık kümesi bölüm ve personel tablolarına dönüştürülür. Bölüm tablosuna personel tablosunun anahtar alanı olan sicilno alanı yabancı anahtar olarak eklenir. İkincisi, benzer şekilde her iki varlık kümesi bölüm ve öğrenci tablolarına dönüştürülür. Öğrenci tablosuna bölüm tablosunun anahtar alanı olan bölümno alanı yabancı anahtar olarak eklenir.



Şekil: Personel-Bölüm yönetir bire-bir ilişkisi ER şeması

ER şemasının tablolara dönüştürme

1. Yol

Bölüm (bölümNo, bölümAd, sicilNo)

Personel (sicilNo, ad, soyad, adres, maaş)

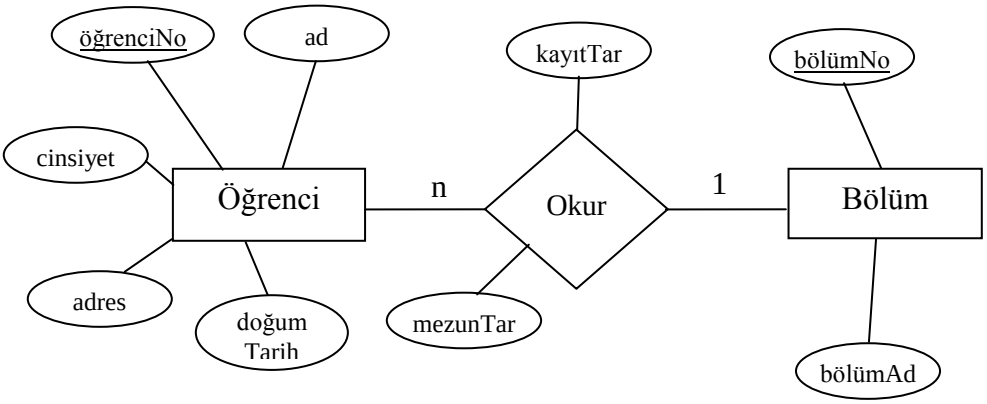
2. Yol

Bölüm (bölümNo, bölümAd)

Personel (sicilNo, ad, soyad, adres, maaş, bölümNo)

Bire-birçok(1-n) ilişkilerin tablolara dönüştürülmesi

İlişkiyi oluşturan varlık kümeleri tablolara dönüştürülür. İlişkinin n tarafındaki tabloya 1 tarafındaki tablonun anahtar alanı yabancı anahtar olarak eklenir. İlişkide belirtilen tanımlayıcı nitelikler n tarafına alan olarak eklenirler. Örneğin öğrenci bölüm varlık kümeleri arasındaki 1-n okur ilişkisinde öğrenci ve bölüm varlık kümeleri tablolara dönüştürülür . İlişkinin n tarafındaki tablo olan öğrenci tablosuna bölüm tablosunun anahtar alanı olan bölümNo alanı yabancı anahtar olarak eklenir. İlişkiyi tanımlayan kayıtTar ve mezunTar nitelikleri de öğrenci tablosuna alan olarak eklenirler.



Şekil: Öğrenci ve Bölüm 1-n okur ilişkisi ER şeması

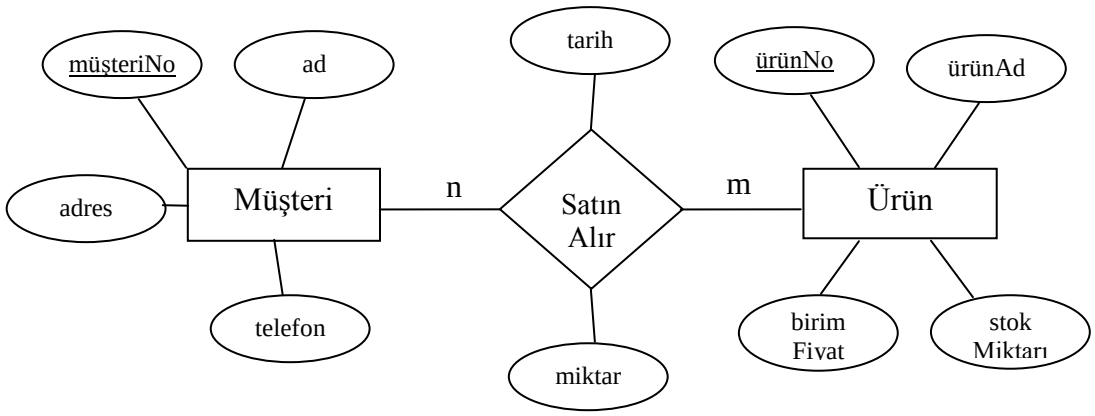
Yukarıdaki ER şeması tablolara dönüştürülürse;

Öğrenci(öğrenciNo, ad, cinsiyet, adres,doğumTarih, bölümNo, kayıtTar, mezunTar)

Bölüm(bölümNo, bölümAd)

Birçoğa-birçok(n-m) ilişkilerin tablolara dönüştürülmesi

İlişkiyi oluşturan varlık kümeleri tablolara dönüştürülür. Ancak ilişki isminde yeni bir tablo oluşturulur. İlişkiyi oluşturan tabloların anahtar alanları yeni tabloya yabancı anahtar olarak eklenir. İlişkide belirtilen tanımlayıcı nitelikler varsa yeni tabloya alan olarak eklenir. Yeni tablonun anahtar alanı ilişkiyi oluşturan tabloların yabancı anahtarlarından oluşan ikili veya daha fazla alandan oluşur. Ancak bazı durumlarda bu anahtar tanımı yeterli olmaz. Yeni tabloya tabloya uygun şekilde yeni bir anahtar alan eklenir. Örneğin aşağıdaki şekildeki müşteri-ürün n-m ilişkisi tabloya dönüştürülürken, müşteri ve ürün adında iki tablo oluşturulur. Ayrıca satınalır isminde yeni bir tablo oluşturulur. Satınalır tablosun müşteri tablosunun anahtar alanı olan müşteriNo ve ürün tablosunun anahtar alanı olan ürünNo yabancı anahtar olarak eklenirler. Ayrıca satın alır ilişkisinde tanımlanan tarih ve miktar nitelikleri de satınalır tablosuna alan olarak eklenirler. Satınalır tablosunun anahtar alanı olarakta satınalırNo isminde yeni bir alan eklenir.



Şekil: Müşteri ürün n-m satın alır ilişkisi ER şeması

Yukarıdaki ER şeması tablolara dönüştürülürse;

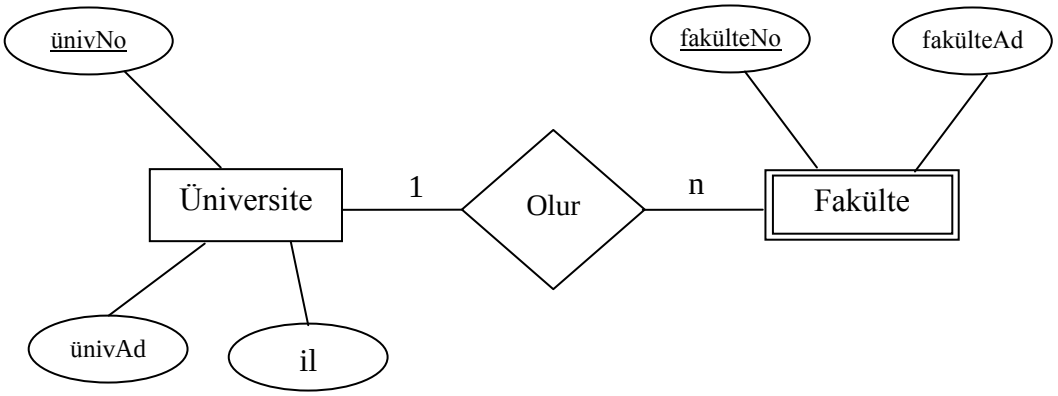
Müşteri (müşteriNo, ad, adres, telefon)

Ürün (ürünNo, ürünAd, birimFiyat, stokMiktarı)

SatınAlır (satınAlırNo, müşteriNo, ürünNo, tarih, miktar)

Zayıf varlık kümelerinin tablolara dönüştürülmesi

Zayıf varlık kümeleri tablolara dönüştürülür. Ancak zayıf varlık kümesinin anahtar alanı kendi anahtar alanı ile birlikte ilişki kurduğu güçlü varlık kümesinin yabancı anahtarının birleşimidir. Örneğin, aşağıdaki şekildeki üniversite-fakülte 1-n ilişkisinde, üniversite varlık kümesi güçlü, fakülte varlık kümesi ise zayıf varlık kümesidir. Üniversite ve fakülte varlık kümeleri, ayrı ayrı üniversite ve fakülte varlık kümelerine dönüştürülür. Ancak, fakülte varlık kümesinin anahtar alanı, fakülteNo ile birlikte üniversiteNo alanlarından oluşturulur.



Şekil : Üniversite-Fakülte olur 1-n güçlü-zayıf ilişkisi

Yukarıdaki ER şeması tablolara dönüştürülürse;

Üniversite (ünivNo, ünivad,il)

Fakülte (fakülteNo, ünivNo, fakülteAd)

ÖDEV:

Muhtarlık, Eczane, Mağaza, Hastane, Okul, Personel, Stok Takip vb konularda geliştirilecek bir otomasyon için ER diyagramlarının detaylı bir şekilde oluşturulması(Varlıkların ve niteliklerinin ve varlık kümeleri arasında ilişkilerin detaylıca tanımlanması vb…), ER diyagramlarından veritabanı tablolarının oluşturulması(Tabloların oluşturulması, niteliklerinin belirlenmesi, tabloların anahtarlarının oluşturulması, tablolar arasındaki ilişkilerin tespiti)…(Teslim Tarihi:24 nisan 2009 Cuma)

SQL
(Structed Query Language)

1. SQL nedir?

İlişkisel Veritabanı Yönetim Sistemleri (Relational Database Management Systems- RDBMS) modeli ilk önce 1970 yılında Dr. E.F. Codd tarafından tarif edilmiştir. SQL veya Structured English Query Language (SEQUEL), IBM firması tarafından Codd'un modelini kullanmak için geliştirilmiştir. SEQUEL daha sonra SQL olmuştur. 1979 yılında, Relational Software (şu an Oracle Corporation), SQL' in ilk ticari uygulamasını geliştirmiştir. Bugün SQL, ilişkisel veritabanı yönetim sistemleri standardı olarak kabul edilmektedir.

SQL (Structured Query Language) kendisi bir programlama dili olmamasına rağmen bir çok kişi tarafından programlama dili olarak bilinir. SQL herhangi bir veri tabanı ortamında kullanılan bir alt dildir (sub language). SQL ile yalnızca veri tabanı üzerinde işlem yapabiliriz. SQL cümlecikleri kullanarak veri tabanına kayıt ekleyebilir, olan kayıtları değiştirebilir silebilir ve bu kayıtlardan listeler oluşturabiliriz.

Veritabanı içindeki verileri elde etmek ve değiştirmekle ilgili SQL deyimleridir.

- SELECT → Verileri istediğimiz şekilde seçebilmemizi sağlar.
- INSERT → Tablo içine yeni kayıt eklememizi sağlar.
- UPDATE → Tablodaki bilgileri değiştirmemizi sağlar
- DELETE → Kayıt yada kayıtları silmemizi sağlar.

SQL'in en önemli ve en çok kullanılan deyimi SELECT deyimidir.

2. Verilere Erişmek (SELECT)

Veritabanındaki verilere erişmenin, diğer bir deyişle onları görmenin en sık kullanılan yöntemidir. Veritabanımızdaki hangi tablo yada tablolardaki alanları (bilgileri) görmek istiyorsak SQL cümlemizin başında mutlaka SELECT deyimi yer almalıdır.

Bu komut ile veritabanı üzerindeki tablonun hangi kolonları alacağımız veritabanına söyleriz. Tablonun bütün kolonlarını görmek istiyorsak '*' karakterini kullanırız. Sadece belli kolonları görmek istiyorsak kolon isimlerini aralarına virgül koyarak yan yana yazmalıyız

SELECT deyiminin Kullanım şekli şöyledir:

SELECT Alan_Adı FROM Tablo_adı;

Aşağıdaki örnekleri açıklayacak olursak birinci örnek personel tablosundaki bütün kayıtları getirecektir.

Örnek 1 :

```
SELECT * FROM personel;
```

Örnek 2 :

```
SELECT * FROM meslekler ;
```

Bütün meslek bilgilerini almak istersek yukarıdaki gibi bir query (sorgu cümlesi) yazmalıyız.

Örnek 3 :

```
SELECT ad, soyad FROM personel ;
```

Bu query ise bize firmamızda çalışan bütün personeli listeler.

FROM

Bu komut bütün SQL cümleciklerinde bulunması gerekli bir komuttur. Bu komut ile hangi tablolar üzerinde çalışacağımızı veri tabanına söylüyoruz. Eğer aynı sql cümleciği ile bir kaç tablo üzerinde işlem yapmak istersek tablo isimleri arasına virgül koymalıyız.

SELECT * FROM personel;
Personel tablosundaki bütün kayıtları seç(göster).

WHERE

Gerçek hayat'ta bu kayıtların sadece bir kısmına ihtiyaç duyarız. Bize gerekli olan dataları diğerlerinden ayıran bazı özellikleri vardır. İşte bu özellikleri Where komutu yardımı ile kullanarak gerekli datalara ulaşabiliriz. Where komutu ile select sorgu cümleciğimize şart ifadesi eklemiş oluyoruz. Şu özelliğe sahip olanları getir şeklinde.

Örnek 4 :

Adı Ahmet olan personeli listelemek istersek ne yapacağız. Aşağıdaki gibi bir sorgulama yapacağız.

```
SELECT * FROM personel WHERE ad='Ahmet';
```

Örnek 5 :

Yaşı 40'dan büyük personeli listeleme istersek;

```
SELECT * FROM personel WHERE Dogum_tarihi < '13.04.1959'
```

sorgulamasını kullanmalıyız. Elemanın 40 yaşında büyük olması için 1959 yılından önce doğmuş olması gerekmektedir. O halde dogum_tarihi 1959 yılından küçük olmalıdır.

Örnek 6 :

Adana'da doğmuş personeli listelemek istersek

```
SELECT * FROM personel WHERE Dogum_yeri = 'Adana'
```

Örnek 7 :

300 milyondan fazla maaş alan kişileri işe maaşa göre sıralamak istersek;

```
SELECT * FROM ucretler WHERE  aylik_ucret >= 300000000 ORDER BY
aylik_ucret;
```

BETWEEN (ARASINDA)

Aralıklı sorgulama yapmak istersek kullanabileceğimiz bir operatördür.

örnek 1 : Öğrenci numarası 240 ile 400 arasında olan öğrenciler kimlerdir?

```
SELECT *
FROM öğrenciler
WHERE öğrenci_no> =240 AND öğrenci_no < = 400;
```

BETWEEN komutu ile daha kısa olacaktır.

```
SELECT *
FROM öğrenciler
WHERE öğrenci_no BETWEEN 240 AND 400;
```

Örnek 2 :

```
SELECT * FROM ucretler WHERE aylik_ucret BETWEEN 200000000 AND
300000000
```

Bu cümlecik ile 200 ile 300 milyon arasında maaş alanlar listelenecektir.

ORDER BY (SIRALA)

Bu komut ile belirtilen kolona göre artan veya azalan bir sıralama ile sorgulama yapabiliriz.

ASC : kullanarak küçükten büyüğe doğru artan sıralama yapabiliriz.

DESC : kullanarak büyükten küçüğe doğru azalan sıralama yapabiliriz.

Ancak ASC kullanmak zorunlu değildir. Çünkü default sıralama tipi ASC'dir. Aynı anda birkaç kolon üzerinden de sıralama yapabiliriz.

Örnek 1 : Öğrenciler tablosundan ,öğrenci_no,adı,soyadı sütunlarını seç ve öğrenci numarasına göre büyükten küçüğe sırala.

```
SELECT öğrenci_no, adı, soyadı
FROM öğrenciler
ORDER BY öğrenci_no DESC;
```

ASC Küçükten büyüğe sıralar, DESC Büyükten küçüğe sıralar.

Bir tablo içinde , birden fazla sütundan aynı anda sıralamak için ;

örnek 2 : Öğrenciler tablosundan seçili sütunları öncelik adda olmak üzere büyükten küçüğe,soyadı büyükten küçüğe ve öğrenci numarasını küçükten büyüğe sıralayalım.

```
SELECT öğrenci_no,adı,soyadı
FROM öğrenciler
ORDER BY adı DESC, soyadı DESC, öğrenci_no ASC;
```

veya;

```
SELECT öğrenci_no,adı,soyadı
FROM öğrenciler
ORDER BY adı, soyadı DESC,öğrenci_no;
```

DESC ' li durumda yanına yazıp belirtilir, yazılmazsa ASC direk kabul edilir.

Örnek 3 :

```
SELECT * FROM personel ORDER BY ad ASC;
```

Bu query ile personel tablosundaki bütün kayıtları ad'a göre küçükten büyüğe doğru sıralarız.

Örnek 4 :

```
SELECT * FROM personel ORDER BY soyad DESC;
```

Bu query da yukarıdakinin tersine kayıtları büyükten küçüğe doğru sıralar.

Örnek 5 :

```
SELECT * FROM personel ORDER BY ad, soyad;
```

Bu query kayıtları ad göre artan bir sıralama yapar. Ancak aynı ad ile yaratılmış birden fazla kayıt varsa ise bunları da soyad sırasına göre artan bir şekilde sıralar. Eğer her iki kolonda aynı ise o zaman okuduğu sırada sıralar.

Örnek 6 :

```
SELECT * FROM personel ORDER BY dogum_tarihi DESC,ad,soyad ;
```

Bu query'de ise personel kayıtları büyükten küçüğe doğru sıralanıyor. Yani en genç eleman'dan başlanarak en yaşlı elemana doğru bir liste yapılıyor. Doğum tarihleri aynı olanlarda ise ad ve soyad'a göre bir sıralama yapılmaktadır.

LIKE

İçinde belli bir karakter dizisi bulunan datalara (verilere) ulaşmak istersek kullanabileceğimiz bir operatördür.

Örnek 1 :

```
SELECT * FROM personel WHERE adres LIKE '%İstanbul%'
```

Bu sorgulama ile adres alanında İstanbul geçen kayıtları listelemiş oluruz.

Örnek 2 :

```
SELECT * FROM personel WHERE adres LIKE '%İstanbul'
```

Bu sorgulama ile adres alanının sonunda İstanbul geçen kayıtları listelemiş oluruz.

Örnek 3 :

```
SELECT * FROM personel WHERE adres LIKE 'İstanbul%'
```

Bu sorgulama ile adres alanının başında İstanbul geçen kayıtları listelemiş oluruz

MAX

Tablo içinde, verilen kolondaki en büyük değeri geri döndürür. Genel yazım biçimi aşağıdaki gibidir;

Select MAX(kolon_adı) FROM tablo;

Örnek 1:

En fazla aylık ücret alan personel ne kadar maaş alıyor?

```
Select MAX(aylik_ucret) From ucretler;
```

Örnek 2:

En fazla aile yardımı alan personelin maaşını ve sicil no' sunu öğrenmek istersek ;

```
Select per_sicil_no,MAX(aylik_ucret) From ucretler ;
```

örnek 3: Okul içerisinde en fazla devamsızlığı bulunan öğrenciyi görmek istersek;

```
SELECT MAX (devamsızlık)
FROM öğrenciler;
```

MIN

Tablo içinde, verilen kolondaki en küçük değeri geri döndürür. Genel yazım biçimi aşağıdaki gibidir;

Select MIN(kolon_adı) FROM tablo;

Örnek :

En düşük aylık ücret alan personel ne kadar maaş alıyor ?

```
Select MIN(aylik_ucret) From ucretler;
```

Örnek :

En az aile yardımı alan personelin maaşını ve sicil no'sunu öğrenmek istersek ;

```
Select per_sicil_no,MIN(aylik_ucret) From ucretler ;
```

SUM (TOPLA)

Verilen kolondaki bütün değerleri toplayarak geri döndürür. Genel yazım biçimi aşağıdaki gibidir;

Select SUM(kolon_adı) FROM tablo;

Örnek 1:

Personele ödenen toplam ücret nedir ?

```
Select SUM(aylik_ucret) From ucretler;
```

örnek 2: Okuldaki öğrencilerin devam etmedikleri günlerin toplamı ne kadardır? dersek;

SELECT SUM (devamsızlık)
FROM öğrenciler;

AVG (ORTALAMA)

Verilen kolondaki değerlerin aritmetiksel ortalamasını geri döndürür. Genel yazım biçimi aşağıdaki gibidir;

Select AVG(kolon_adı) FROM tablo;

Örnek :

Aylık ödenen ücret ortalamasını bulmak istersek ;

```
Select AVG(aylik_ucret) From ucretler;
```

SQL DEYİMLERİ VE ÖRNEKLERİ:

CREATE DEYİMİ

CREATE deyimi tablo ve view gibi bir veritabanı nesnesi yaratmayı sağlar. Veritabanı üzerinde bir tablo veya bir veritabanı nesnesi yaratmak için CREATE deyimi kullanılır.

Yapısı:
CREATE <tablo adı>
Örnek:

```
CREATE TABLE Musteri
(
  mus_id      char(4)       NOT NULL
  mus_ad      varchar(40)   NULL,
  ili         varchar(20)   NULL,
  ulke        char(2)       NULL,
  adres       varchar(30)   NULL
)
```

NOT: Char, varchar, integer, numeric gibi sözcükler tablo alanlarındaki temsil edilecek verinin türünü belirtir. SQL'de SMALLINT, VARCHAR, DECIMAL(x,y), FLOAT(x;y), DATE, LOGICAL, TIME, TIMESTAMP, GRAPHIC(n) gibi alan veri türleri vardır.

Örnek:

```
CREATE TABLE  Personel
(
```

```
Sskno Integer,
Adi Varchar(20) not null,
Soyadi Varchar(20) not null,
Departman integer
)
```

Örnek:

CREATE DATABASE Person

Person adında bir veri tabanı oluşturulur.

Örnek:

```
CREATE TABLE PERSONEL(
PERSONEL_ID int,
AD varchar(10),
SOYAD varchar(10)
)
```

Bu şekilde bir yazımla PERSONEL adında bir tablo oluşturulur. Tablo sütunları da PERSONEL_ID, AD, SOYAD'dır.

Örnek:

```
CREATE TABLE PERSONELYAKIN(
PERSONEL_ID int,
YAKIN_ID int,
YAKIN_AD varchar(10),
YAKIN_SOYAD varchar(10)
)
```

Bu örnekte de PERSONELYAKIN adında bir tablo oluşturulmuştur. Tablo sütunları da PERSONEL_ID, YAKIN_ID, YAKIN_AD, YAKIN_SOYAD' dır.

ALTER DEYİMİ

Daha önce yaratılmış nesnenin değiştirilmesini sağlar. Örneğin bir tablonun tasarımını değiştirmek gibi.

Örnek:

```
ALTER TABLE Musteri
ADD tel varchar(20) NOT NULL
```

Yukarıdaki deyimde musteri tablosunun alanlarına tel adlı bir alan daha eklenmiştir.

DROP DEYİMİ

Bir nesnesin silinmesini sağlar.

Örnek:

```
DROP TABLE MUSTERI
```

Müşteri tablosunun verilerini ve tabloyu siler.

SELECT DEYİMİ

Veritabanındaki verilere erişmenin, diğer bir deyişle onları görmenin ya da onları elde etmenin en sık kullanılan yöntemidir. Genellikle bir ya da daha çok tablonun bütün alanları ya da belli alanları için SELECT deyimi yazılır.

Temel Yapısı:

SELECT [ALL] [DISTINCT] liste [INTO yeni tablo] FROM [tablo]
[WHERE ifade]
[GROUP BY ifade]
[HAVING ifade]
[ORDER BY ifade]
[COMPUTE ifade]

Seçeneklerin Anlamları:

ALL sözcüğü bütün satırların sonuç listesinde görünmesini sağlar.
DISTINCT sözcüğü sadece tek olan (unique) kayıtların sonuç listesinde yer almasını sağlar.
liste parametresi veriden seçilecek kolonu (sütunu) belirtir.
INTO sözcüğü yeni bir tablo yaratmayı sağlar.
yeni tablo parametresi sorgu sonucu yaratılacak tabloyu belirtir.
FROM sözcüğü belli bir tablonun seçilmesini sağlar.
tablo parametresi ise sorgulanacak olan tablo ya da tabloları, görünümleri belirtir.
WHERE bir koşulu belirterek sadece o kuşula uyan kayıtların seçilmesini sağlar.
GROUP BY Kayıtların gruplanmasını sağlar. HAVING deyimiyle de ara toplamların alınmasını sağlar.
HAVING sözcüğü de kayıtlarda kısıtlama yapar ancak hesaplamayı etkilemez.
ORDER BY sözcüğü ise belirtilen kolona göre listelenen kayıtları sıralamayı sağlar. Sıralama artan (ASC) ya da azalan (DESC) olabilir
COMPUTE sözcüğü ise hesaplama yapar. Tipik olarak SUM, AVG, MIN, MAX, COUNT gibi fonksiyonları kullanarak hesaplama yapar.

Örnek:

SELECT * FROM musteri

Yukarıdaki deyim ile musteri tablosundaki bütün bilgiler elde edilir. SELECT deyiminin ardından kullanılan * (asterisk) işareti bütün kayıtlar anlamına gelir. Bu deyimin aynısı (aynı sonucu vereni) şu şekilde de yapılabilir:

Örnek:

SELECT kodu, ad, soyad, grup, il, bakiye FROM musteri

Müşteri tablosu:

kodu	Ad	Soyad	grup	İl	bakiye
1	Ahmet	Uzun	ithal	İZMİR	300000
2	Ayşe	Yılmaz	ithal	ANKARA	400000

3	Mehmet	Yılmaz	ihraç	ANKARA	100000
4	Hüseyin	Uzun	İhraç	İZMİR	600000
5	Nuri	Gezer	İthal	İZMİR	900000
6	Fatma	Örnek	İhraç	İSTANBUL	300000

Sorgunun Sonucu:

1	Ahmet	Uzun	İthal	İZMİR	300000
2	Ayşe	Yılmaz	İthal	ANKARA	400000
3	Mehmet	Yılmaz	İhraç	ANKARA	100000
4	Hüseyin	Uzun	İhraç	İZMİR	600000
5	Nuri	Gezer	İthal	İZMİR	900000
6	Fatma	Örnek	İhraç	İSTANBUL	300000

SELECT deyimi ile sadece belli kolonlar (alanlar) da seçilebilir:
Arama Kriterleri:

Karşılaştırma operatörleri	(=, >, <, >=, <>, !=, !<, !>
Aralık belirtme	BETWEEN ve NOT BETWEEN
Liste	IN ve NOT IN
String karşılaştırma	LIKE ve NOT LIKE
Bilinmeyen değerler	IS NULL ve IS NOT NULL
Koşulların birleştirilmesi	AND, OR
Olumsuzlaştırma	NOT

ÖRNEK:

- SELECT * from musteri WHERE bakiye BETWEEN 100000 AND 3000000

- SELECT * from musteri WHERE bakiye <= 100000 AND >= 3000000

1	Ahmet	Uzun	İthal	İZMİR	300000
3	Mehmet	Yılmaz	İhraç	ANKARA	100000
6	Fatma	Örnek	İhraç	İSTANBUL	300000

ÖRNEK:

```
SELECT * FROM MUSTERI

WHERE bakiye BETWEEN 100000 AND 500000

AND grup='ithal'
```

ARİTMETİK İŞLEMLER

Aritmetik işlemleri gerçekleştirmek için belli operatörler kullanılır:

OPERATÖRLER AÇIKLAMALARI

+	Toplama
-	Çıkarma
/	Bölme
*	Çarpma

ÖRNEK:

SELECT bakiye, bakiye*2 FROM MUSTERI

VERİLERİ SIRALAMAK

SELECT deyimi ile elde edilen veriler istenirse sıralanabilir. Sıralama belirtilen bir ya da daha fazla kolona göre yapılır. Bunun dışında sıralama ASC (ascending-artan) ya da DESC (descending-azalan) olarak belirtilebilir.

Kullanım Biçimi:

SELECT kolon_listesi

ORDER BY kolon adı **ASC** ya da **DESC**

ORDER BY sözcüğü ise verilerin istenilen alan göre sıralı olarak listelenmesini sağlar.

ÖRNEK:

SELECT * FROM musteri

ORDER BY ad

Yukarıdaki deyim ile müşteri tablosundaki bütün kayıtlar ad alanına göre sıralı olarak listelenirler.

INSERT DEYİMİ

Tabloya veri girmek için kullanılır.

INSERT INTO <tablo adı>
(sütunlar listesi**) VALUES (**değerler listesi**)**

ÖRNEK:

INSERT INTO CARIANA

(kodu, adi, grubu, adresi)

VALUES ('600', 'FARUK', 'A', '76 sokak no 5')

ÖRNEK:

INSERT INTO MUSTERI

(kodu,ad,soyad,grup,il,bakiye) VALUES

(‘05’,’Muhammet’,’BAYKARA’,’ihraç’,’Elazığ’,’500000’)

UPDATE DEYİMİ:

Tablodaki verileri güncellemek için kullanılır. Genellikle güncelleştirilecek satırı belirtmek için WHERE sözcüğüyle kullanılır.

Mevcut bir tablodaki satırları değiştirmek için UPDATE deyimi kullanılır. UPDATE deyimi sadece bir tablo üzerinde kullanılmalıdır. UPDATE deyimi ile SET ve WHERE sözcüğü kullanılır.

SET sözcüğü değiştirilecek kolonları ve değerleri belirtir. WHERE sözcüğü ise değiştirilecek satırı belirtir.

Kullanım biçimi:

UPDATE *tablo*

SET *kolon = ifade*

WHERE *arama_koşulu*

Örnek: Aşağıdaki örnekte fiyat değerini %10 artırır.

UPDATE siparis

SET fiyatı= fiyatı * 1.1

Örnek:

UPDATE MUSTERI

SET bakiye=bakiye *1.5

Where ad=’Muhammet’

Örnek:

UPDATE Musteri

SET Ad = 'Nuri Yılmaz'

WHERE kod='1';

DELETE DEYİMİ

Bir tablodaki verileri silmek için DELETE komutu kullanılır. Örneğin Öğrenci tablosundaki tüm verileri silmek için;

DELETE * from musteri;

Tabloda, bakiyesi 1000'den küçük olan müşterilerin satırlarını silmek için:

DELETE * FROM musteri WHERE bakiye <=1000

Kullanım biçimi:

DELETE tablo

WHERE arama_koşulu

Örnek: Tablodan satır silmek

Aşağıdaki örnekte müşteri tablosundan ithal grubuna sahip olan müşteriler silinir.

DELETE musteri

WHERE grup = 'ithal'

“

Örnek:

DELETE MUSTERI

WHERE ad=’Muhammet’

Örnek:

DELETE MUSTERI

WHERE bakiye<100000