

Yazılım Kalite Güvencesi

- Yazılım belirlenen bütçe ile zamanında bitirilebilir, kabul edilebilir düzeyde hatasız, planlanan şekilde dağıtılabilen, gereksinimleri karşılayabilen ve sürdürülebilir özelliklere sahip yazılımlar olarak değerlendirilebilir.
- Tüm hataları oluşmadan önlemek, proje koşulları ve maliyetleri içerisinde olanaklı değildir.
- Yazılımda ürünün kalitesini, ağırlıklı olarak üreten sürecin kalitesi belirlemektedir. Bu nedenle, yazılım sektöründe ürün odaklı kalite yönetimi anlayışından çok süreç odaklı kalite yönetimi anlayışı hâkimdir.

Temel Kalite Kavramları

- **Kalite Kontrol:** Ürün veya hizmetin, belirlenen kalite standartlarına uygun olup olmadığını kontrol etmek için yapılan test ve denetim sürecidir.
- **Kalite Güvencesi:** Ürün veya hizmetin, baştan sona kalite standartlarına uygun olmasını sağlamak için süreçleri planlayan ve yöneten proaktif bir yaklaşımdır.
- **Kalite Yönetimi:** Kalite hedeflerine ulaşmak için süreçleri, politikaları ve sorumlulukları bir bütün olarak yöneten sistematik bir yaklaşımdır.
- **Kalite Sistemi:** Bir organizasyonun kalite hedeflerine ulaşmasını sağlamak için kullanılan yapı, süreçler ve araçların bütünüdür.
- **Kalite Denetimi:** Bir ürünün, hizmetin veya sürecin belirlenen kalite standartlarına uyup uymadığını bağımsız bir şekilde değerlendirme sürecidir.
- **Kalite İyileştirme:** Mevcut süreçlerin veya ürünlerin performansını artırmak ve kalite standartlarını yükseltmek için yapılan sürekli geliştirme faaliyetleridir.

Yazılımın Kalitesini Belirleyen Özellikler

- Yazılımın çalışmasına yönelik
- Yazılımın gelişmesine yönelik
- Yazılımın uyumluluğuna yönelik

Yazılımın Çalışmasına Yönelik Özellikler

- Doğruluk
- Güvenilirlik
- Verimlilik
- Korunmuşluk
- Kullanılabilirlik

Yazılımın Gelişmesine Yönelik Özellikler

- Bakım kolaylığı
- Esneklik
- Sınanabilirlik

Yazılımın Uyumluluğuna Yönelik Özellikler

- Taşınabilirlik
- Tekrar kullanılabilirlik
- Uyumluluk

Yazılım Kalitesini Değerlendiren Modeller ve Standartlar

- 1980'lerin ortalarından sonra yazılımda kalite sertifikasyonu, süreç iyileştirme ve yetenek belirleme amaçlarıyla çeşitli modeller ortaya çıkmıştır.
 - ISO 9000-3
 - ISO/IEC 12207
 - ISO/IEC 25010
 - ISO 15504 (SPICE)
 - TickIT
 - Trillium
 - CMMI (Capability Maturity Model Integration)
 - Agile ve DevOps Prensipleri

- **ISO 9000-3**, yazılım sektörüne yönelik olarak geliştirilmiş ISO 9000 kalite yönetim standartları serisinin bir rehberidir. Bu standart, yazılım geliştirme süreçlerine kalite yönetimi uygulamak için yönergeler sunar. Genel olarak, yazılım ürünlerinin ve süreçlerinin kalitesini artırmak amacıyla kullanılır.
- **ISO/IEC 12207**, **yazılım yaşam döngüsü** süreçlerini tanımlayan bir standarttır. Yazılım geliştirme, teslimat, bakım ve yönetim süreçlerini yapılandırmak için bir çerçeve sunar. Yazılım geliştirme sürecinin düzenlenmesi ve optimize edilmesi gerektiğinde kullanılır
- **ISO/IEC 25010**, **yazılım ürünlerinin kalite** özelliklerini değerlendirmek için kullanılan bir kalite modeli standardıdır. Bu standart, yazılım ürünlerinin kalitesini anlamak ve değerlendirmek için bir çerçeve sunar. Yazılım ürününün kalite seviyesini ölçmek ve değerlendirmek istediğinizde kullanılır.

- **ISO/IEC 15504 SPICE** (Software Process Improvement and Capability Determination), yazılım geliştirme süreçlerinin olgunluk seviyesini değerlendirmek ve sürekli iyileştirmek için kullanılan bir standarttır. ISO/IEC 12207 gibi yaşam döngüsü süreçlerini tanımlayan standartlarla yakından ilişkilidir. Bu standart, yazılım süreçlerinin değerlendirilmesi ve iyileştirilmesine yönelik bir çerçeve sunar.
- **TickIT**, İngiltere ve Norveç yazılım geliştiricileri tarafından özellikle yazılım geliştirme ve BT (Bilgi Teknolojisi) hizmetleri alanında kalite yönetim sistemlerinin uygulanmasını ve denetlenmesini desteklemek için geliştirilmiş bir kalite yönetim sistemi sertifikasyon programıdır.

- **Trillium**, yazılım süreçlerinin olgunluğunu değerlendirmek ve iyileştirmek için kullanılan bir kalite modeli ve metodolojisidir. Özellikle telekomünikasyon sektöründe yazılım süreçlerini iyileştirmek amacıyla geliştirilmiştir. Trillium, yazılım geliştirme süreçlerinin etkinliğini ve kalitesini artırmaya yönelik bir rehber sunar.
- **CMMI** (Capability Maturity Model Integration), bir organizasyonun süreçlerini değerlendirmek, olgunlaştırmak ve iyileştirmek için kullanılan bir çerçeve ve modeldir. CMMI, özellikle yazılım geliştirme, sistem mühendisliği, proje yönetimi ve ürün geliştirme gibi alanlarda süreçlerin etkili bir şekilde yönetilmesine odaklanır.
- **Agile** ve **DevOps**, yazılım geliştirme ve teslimat süreçlerini daha hızlı, verimli ve kaliteli hale getirmek için kullanılan iki modern yaklaşımdır. Her ikisi de süreçlerin iyileştirilmesi ve müşteri odaklı çözümler sunma hedefini taşır, ancak farklı odak noktalarına sahiptirler.

CMMI (Tümleşik Yetenek Olgunluk Modeli- Capability Maturity Model Integration) Modeli

- CMMI, her organizasyon için yazılım geliştirme yeteneğini değerlendirmek ve aynı zamanda yazılım kalitesini artırmak için kullanılan bir modeldir. Temelinde bir süreç modeli olup, örgütlerin yazılım süreçlerinin olgunluğunu değerlendirme modelidir.
- CMMI süreç iyileştirme modelidir. Özellikle büyük organizasyonlarda ve karmaşık projelerde, süreçlerin standartlaşması, verimliliğin artması ve kalite seviyesinin korunması için kullanılır.

- Organizasyonların; insan kaynaklarını, süreçlerini ve teknolojilerini geliştirip daha iyi çalışır hale getirebilmelerini sağlayan uzun vadeye yayılmış bir olgunlaştırma planıdır.
- Yazılım süreçlerindeki performans, maliyet ve zaman tahmini gibi konularda yaşadığı sıkıntıları gidermek ve yazılımda kalite sertifikasyonunu sağlamak amacıyla çeşitli süreç iyileştirme ve yetenek belirleme modelleri geliştirilmiştir.
- Örgütlerin yazılım süreçleri (Yazılım planlama, geliştirme, yapılandırma vb.) sırasında kurumlara rehberlik ederek ürün kalitesini artırmaya yardımcı, olgunluğu değerlendirme modelidir.
- Carnegie Mellon Üniversitesi'ne bağlı *Yazılım Mühendisliği Enstitüsü* tarafından Amerikan Savunma Bakanlığı'nın isteği üzerine 1986 yılında geliştirilmeye başlanmıştır. İlk olarak yalnızca yazılım üzerine, SW-CMM olarak 1991 yılında yayınlanan model, 2002 yılında sektör bağımsız bir şekil alarak "CMMI" olmuş, 2006 yılında ise v1.2 olan son sürümüne ulaşmıştır.
- Modeller içerisinde en önemlilerinden biridir.

- Yazılımı değerlendirmek, yazılım kalitesini artırmak ve yazılım süreçlerinin olgunluğunu değerlendirmek için kullanılan bir modeldir.
- Basamaklı ve sürekli olmak üzere iki gösterime sahiptir. İki gösterimde birbirini destekler. Her iki durumda da Süreç Alanları, Hedefler ve Uygulamalar aynıdır. Bu temsiller arasındaki fark, süreç iyileştirmenin uygulanma sırasıdır.

- 1997 yılında farklı alanlara dağılmış olan olgunluk modellerinin tek bir çatı altında toplanmasına ve böylece denetimin daha kolay yapılabileceğine karar verilmiştir. Bu durumu sağlamak için SEI aşağıda belirtilen üç modeli Entegre Yetenek Olgunluk Modeli (CMMI) adı altında birleştirmiştir:
 - Yazılım Yetenek Olgunluk Modeli (SW-CMM)
 - Sistem Mühendisliği Yetenek Olgunluk Modeli (SE-CMM)
 - Entegre Ürün Geliştirme Yetenek Olgunluk Modeli (IPD-CMM)

- CMMI, çeşitli süreç alanlarında iyileştirmeler yapmaya odaklanır. Bu alanlar genellikle şu başlıklar altında toplanır:
 - **Proje Yönetimi:** Proje planlama, proje izleme ve kontrol gibi alanları içerir.
 - **Mühendislik:** Gereksinim yönetimi, tasarım, doğrulama ve geçerlilik gibi mühendislik süreçlerini kapsar.
 - **Süreç Yönetimi:** Organizasyonel süreç odaklanma, süreç tanımlama ve süreç iyileştirme gibi yönetim süreçleriyle ilgilenir.
 - **Destek:** Konfigürasyon yönetimi, kalite güvencesi, risk yönetimi gibi destek süreçlerini içerir.

Yazılım Yetenek Olgunluk Modeli (SW-CMM)

- Organizasyonlardaki yazılım süreçlerinde, bu organizasyonun yazılım konusunda bulunduğu olgunluk seviyesi ve bu seviyeyi yukarı taşıyabilmek için bu organizasyonların sahip olması gereken temel unsurları belirten modeldir. Yazılım olgunluk modelinin genel olarak beş seviyesi bulunmaktadır:

CMMI'nin 5 Seviyesi

1. Başlangıç (Initial): Başarının bireysel gayretlere dayandığı, çok az sayıda sürecin tanımlı olduğu seviyedir. Bu seviye hariç diğer bütün seviyelerde o seviyelerin ilgilendiği belirli konular olmasına rağmen bu seviyede yoktur.
2. Tekrarlanabilir (Repeatable): Bu seviyede, ilk olarak yazılım gereksinimleri yönetilir ve bu gereksinimler ile ilgili ürünler oluşturulur. Daha önceden hizmet verilen organizasyona karşı verilen taahhütler, bu seviyede yine bu organizasyonun istek ve revizyonlarına uygun şekilde tesis edilir, iki taraf arasında gözden geçirilip kontrol edilir.
3. Tanımlanmış (Defined): Bu seviyede, isminden de anlaşılacağı üzere tüm süreçler, artık iyice tanımlanmış standartlar, prosedürler, araçlar ve metodlar ile anlatılmıştır. Ayrıca yazılım ile ilgili olan tüm projelerde tam anlamıyla CMMI standartları uygulanmaya başlanır. Bu seviyedeki tüm süreçler, 2. seviyeye göre daha kapsamlı ve detaylı olarak tanımlanmıştır.
4. Yönetilen (Managed): Bu seviye yazılım süreçlerinin, iyileştirmeler öncesi son şeklini almadan, tam anlamıyla yönetildiği ve tamamlandığı yerdir. Bu seviyede çeşitli istatistiksel ve nicel ölçüm teknikleri kullanılarak süreçlerin performansları kontrol edilir ve bu süreçlerin ileri zamanlardaki performansları öngörülebilir. Bu olay 4. seviyeyi 3. seviyeden ayıran en önemli unsurdur.
5. Optimize Edilen (Optimizing): 4. seviyede elde edilen veriler ve projelerin tamamlandığı süreçte gelişen teknoloji ile bu seviyede sonuçlandırılan tüm projeleri iyileştirmeye ve projelerden maksimum fayda sağlamaya çalışılır. Bu süreçte tüm organizasyon projeleri iyileştirmeye odaklanmıştır.

CMMI'nin 6 Yetenek Düzeyi

0. Incomplete: Sürecin belirlenen hedeflerinden biri ya da daha fazlası başarılamamıştır.

1. Performed: Ürün ortaya çıkarken yapılması gereken iş yapılır ancak bu gelişim kurumsallaşmadığı takdirde zamanla kaybolacaktır.

2. Managed : Süreç planlanır ve plana bağlı kalınır.

3. Defined : Bu seviyeyi bir öncekinden ayıran en büyük fark, standartların kapsamı, süreç tanımlamaları ve uygulanan prosedürlerin işlevselliğidir.

4. Quantitatively Managed : Kalite açısından amaçlanan ölçümlere ve süreç performansına ulaşılmıştır. Bu veriler istatistiksel terimlerle ifade edilir ve süreç devam ettiği sürece inceleme devam eder.

5. Optimizing : Sürekli iyileşen ve gelişen yenilikçi bir süreç performansı gözlenir.

- **Olgunluk Seviyeleri**, tüm organizasyonun süreç yapısının genel gelişmişlik düzeyini ifade eder. Bu, organizasyonun toplamda ne kadar olgun bir süreç yapısına sahip olduğunu gösterir.
- **Yetenek Düzeyleri**, belirli bir süreç alanındaki yetkinliği ölçer. Yani, her bir sürecin kendi başına ne kadar olgun olduğunu ifade eder.

Basamaklı Gösterim

- Başlangıç (initial), yönetilen (managed), tanımlı (defined), nicel yönetilen (quantitatively managed) ve en iyileşen (optimizing) olmak üzere 5 **olgunluk düzeyi** vardır.
- İlk olarak hangi süreçlerin iyileştirileceğini bilmeyen organizasyonlar için en uygun gösterim modelidir ve organizasyonun tümünün olgunluk durumu için uygulanır. Planlama için genel durum ortaya çıkar.
- Her olgunluk seviyesi için uygulanabilir süreç alanları önerir.
- Süreçlerin birbirlerini nasıl desteklediği konusunda yönlendirme sağlar.
- Benchmarking yapılmasını sağlar.

Sürekli Gösterim

- Bir organizasyonun belirli bir süreç alanındaki başarısını ölçmek üzere, 6 yetenek seviyesi kullanır (**Yeterlik Düzeyleri**). Bu yeterlik seviyeleri 0 ile 5 arasında numaralandırılıp; eksik (incomplete), başlangıç (initial), yönetilen (managed), tanımlı (defined), nicel yönetilen (quantitatively managed) ve en iyileşen (optimizing) olarak adlandırılır.
- Sürekli gösterimde CMMI'nın 25 süreç alanı, süreç yönetimi, proje yönetimi, mühendislik ve destek olmak üzere 4 süreç alanı kategorisine ayrılmaktadır.

- Sürekli gösterim, süreç alanları içinden iyileştirmek istediğimiz süreç alanlarının tek tek seçilmesine izin veren bir yaklaşımdır.
- Her süreç alanında yapılacak iyileştirmeleri planlamada yardımcı olur.
- Bir seviyenin tam karşılanmasını sağlayamayacak iyileştirme çalışmalarının izlemesini ve raporlandırılmasına yardımcı olur.
- Bu gösterimi kullanmak için hem kurumun hem de süreç alanlarının ve süreç alanları arasındaki ilişkilerin çok iyi bilinmesi gerekir.
- Bir organizasyonun her bir süreç alanında süreç iyileştirmede gösterdiği başarı için uygulanır.

CMMI AVANTAJLARI

- Tek başına kullanılan diğer süreç iyileştirme ürünlerine göre, ürün yaşam döngüsüne ilişkin daha detaylı bilgi sağlar.
- Organizasyonun süreçlerini iyileştirmesi için esnek ve detaylı değerlendirme olanağı sunmaktadır.
- Tipik olarak organizasyonun farklı kısımlarında yer alan sorunları çözmemiz için fırsat sağlar.
- Esnek yapısından dolayı her boy firma için anlamlı bir değerlendirme modelidir.
- Sistem ve yazılım mühendisliği arasındaki iş birliğini yükseltmeye çalışır, böylece son ürün ve onunla ilintili süreçlere odaklanma şansı sağlar.

Yazılımı Planlama için;

- Kaynaklar belirlenir
- Maliyet kestirimleri yapılır
- Zaman-iş gücü planlaması modelleri oluşturulur
- Proje ekip yapısı oluşturmaya yönelik kavramlar oluşturulur
- Ayrıntılı proje planı oluşturulur
- Proje izlenir

Yazılımı Ölçme

- Ölçme, belli bir nesnenin ya da nesnelerin belli bir özelliğe sahip olup olmadığının gözlenerek sonuçların sayılarla ya da başka sembollerle gösterilmesidir.
 - İş gücü yönetimi
 - Maliyet yönetimi
 - Zaman yönetimi
 - Bilişim sistemleri yönetimi
 - Alım yönetimi
- Ölçme-Kestirme arasında fark vardır.

Yazılım Büyüklüğünü Ölçme

- Uzunluk
 - Doküman (sayfa sayısı, tablo sayısı vb.)
 - Kod (satır sayısı, Halstead ($N = \text{Toplam operatör sayısı} + \text{Toplam değişken sayısı}$))
- işlevsellik
 - İşlev Puanı, özellik puanı vb.
- Karmaşıklık
 - Problem karmaşıklığı, Big-O notasyonu
- Tekrar Kullanım (% olarak)

Yazılımda Kestirim

- Yazılım kestirimi ya da diğer ifadeyle yazılım tahmin işlemi çeşitli yöntemlerle yazılım projelerinin planlaması ve kontrolü için yapılır
- Yazılım geliştirmedeki en zorlu çalışmalardan biridir.

- Yazılımı geliştirmek ne kadar sürecektir?
- Ne kadar efor ve maliyet gerektirecek?
- Kaç geliştirici çalışacak?
- Doğru kaynaklar kullanılıyor mu?
- Maliyet ve risk faktörleri ne?
- Projelerimiz için nasıl güvenilir tahminler elde edebiliriz?
- Uygulama yeterince test edildi mi?

- Yazılım projelerinde tahmin yapılacak 4 ana kıstas bulunur:
 - Yazılım Büyüklük Tahmini / Kestirimi (Size Estimation)
 - Yazılım Efor Tahmini / Kestirimi (Effort Estimation)
 - Yazılım Takvim Tahmini / Kestirimi (Schedule Estimation)
 - Yazılım Maliyet Tahmini / Kestirimi (Cost Estimation)

- Yazılımda büyüklüğü ölçmek için kullanılan bu teknikler:
 - Fiziksel yazılım nesnelerini kullanarak hesaplama (örn. kod satır sayısını kullanma.)
 - Wideband Delphi süreci
 - COCOMO (Constructive Cost Model) modeli
 - User story points – Planning Poker
 - Ürünün işlevsel kısımlarını hesaplama. (alt sistemlerin, sınıfların/modüllerin, yöntemlerin/fonksiyonlarının, ekranların, diyalogların, dosyaların, veri tabanı tablolarının, raporların, mesajların vb. sayısını içerebilir.)
 - IFPUG (ISO/IEC 20926)
 - Mark II (ISO/IEC 20968)
 - NESMA (ISO/IEC 24570)
 - FISMA (ISO/IEC 29881)
 - COSMIC (ISO/IEC 19761)

Proje Maliyetleri

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

Maliyet yönetimi

- Proje teslim zamanı ve gerekli bütçe dahilinde projenin tamamlanması.
- Süreçin kolaylaşması.
- Kaynak kullanımının etkin kullanımı.
- Ürünün doğru fiyatlandırılması.

İşlev Noktaları Yönetimi

- Yazılımın çözümleme aşamasında belirlenen değerdir.
- Üç temel adım izlenir;
 - Problem bilgi ortamının incelenmesi
 - Problem karmaşıklığının incelenmesi
 - İşlev noktası hesaplama

- Kullanıcı Girdileri: personel sicil bilgileri, personel izin bilgileri gibi
- Kullanıcı Çıktıları: her türlü mantıksal çıktı; raporlar, ekran çıktıları, hata iletileri,...
- Kullanıcı Sorguları: personel sicil bilgilerinin sorgulaması, personel maaş bilgilerinin sorgulaması
- Dosyalar: Her türlü mantıksal bilgi yığını, tablolar, veri tabanları
- Dışsal arayüzler: Başka programlarla veri iletimi. import/export

Yukarıda verilen durumlara ait sayılar ağırlık faktörleriyle çarpılarak toplam işlemi yapılır. Çıkan değer Ayarlanmamış İşlev Noktası (AIN) olarak adlandırılır.

Ölçüm Parametresi	Sayı	Ağırlık Faktörü				
		Yalın	Ortalama	Karmaşık		
Kullanıcı Girdi sayısı	?	3	4	6	=	
Kullanıcı Çıktı sayısı	?	4	5	7	=	
Kullanıcı Sorgu Sayısı	?	3	4	6	=	
Kütük Sayısı	?	7	10	15	=	
Dışsal Arayüz Sayısı	?	5	7	10	=	
Toplam Sayı					=	

1. Uygulama, güvenilir yedekleme ve kurtarma gerektiriyor mu?
2. Veri iletişimi gerektiriyor mu?
3. Dağıtılmış İşlemler var mı?
4. Performans kritik mi?
5. Girdiler, çıktılar, dosyalar ya da sorgular karmaşık mı?
6. İçsel işlemler karmaşık mı?
7. Tasarlanacak kod yeniden kullanılabilir mi?
8. Dönüştürme ve kurulun tasarımında dikkate alınacak mı?

14.

Cevaplar 0 ile 5 arasında puanlandırılır

Bunlar hesaplanıp toplanarak Teknik Karmaşıklık Faktörü (TKF) elde edilir.

- $\dot{I}N = A\dot{I}N * (0,65 * 0,01 * TKF)$

Değişik amaçlarla kullanılabilir

- Üretkenlik = $\dot{I}N / \text{Kişi-Ay}$
- Kalite = $\text{Hatalar} / \dot{I}N$
- Maliyet = $\$ / \dot{I}N$

Assembly	300
Cobol	100
Fortran	100
Pascal	90
C	90
Ada	70
Nesne Kökenli Diller	30
4. Kuşak Dilleri	20
Kod Üreticiler	15

COCOMO (Constructive Costing Model- Etkin Maliyet Modeli)Modeli

- 1981 yılında, Barry Boehm tarafından geliştirilmiş algoritmik bir yazılım maliyet kestirim yöntemidir.
- Geçmiş proje verileri ve mevcut proje özelliklerinden türetilen parametreler ile beraber temel bir regresyon formülü kullanır.
- Mikro modeldir.
- Uygulama, ayrıntı düzeyine göre üç ayrı model biçiminde olabilir;
 - Temel (basit) model
 - Ara (orta) model
 - Ayrıntı (detaylı) model
- Ayrıca bu modellerde kullanılacak problemler üç sınıf altında toplanmıştır,
 - Organik (Ayrık),
 - Yarı ayrık
 - Gömülü

- Temel model de girdi satır sayısı kestirimi, temel çıktı ise iş gücü ve zamandır. (Satır değeri bin türündendir)
 - İş gücü (K) $K = a \times \text{Sat}^b$
 - Zaman (T) $T = c \times K^d$
 - Adam-ay = K / T
 - Ayrık Projeler İçin: $a=2.4, b=1.05, c=2.5, d=0.38$
 - Yarı – Gömülü Projeler İçin: $a=3.0, b=1.12, c=2.5, d=0.35$
 - Gömülü Projeler İçin: $a=3.6, b=1.20, c=2.5, d=0.32$

- Ara modelde;
 - Ayrık projeler: $K = 3.2 \times S^{1.05}$
 - Yarı-gömülü projeler: $K = 3.0 \times S^{1.12}$
 - Gömülü Projeler: $K = 2.8 \times S^{1.20}$

- 50.000 satır kod için, *Orta Ölçekli* tasarlanıyor. Bu durumda Cocomo modelinin temel denklemlerini kullanarak proje süresi ve iş gücü;
 - İş gücü (K) $K = 3.0 \times 50^{1.12} = 239,87$ adam-ay
 - Zaman (T) $T = 2.5 \times K^{0.35} = 17.02$ ay
- Eğer projede **1 kişi** çalışacaksa, proje yaklaşık **239,87 ay** sürer.
- 17 ay, Cocomo modeline göre projenin toplam tamamlanma süresidir. Bu süre, proje için ideal ekip büyüklüğü ve verimlilik göz önüne alınarak hesaplanmıştır. Yani, projenin 239,87 adam-aylık iş gücü gereksinimi 17 aya yayılırsa, ekip doğru bir şekilde çalışarak projeyi bu sürede tamamlayabilir.

- Cocomo modelinde maliyet çarpanı (C), projenin toplam maliyetini etkileyen faktörleri göz önüne alarak kullanılan bir katsayıdır. Bu çarpan, projenin karmaşıklığı, personelin yetenek seviyesi, geliştirme ortamı, gereksinimlerin netliği gibi çeşitli faktörleri kapsar. COCOMO'nun daha gelişmiş versiyonlarında (COCOMO II gibi), maliyet çarpanı genellikle **Çarpan Faktörleri (Cost Drivers)** olarak adlandırılır ve her bir faktör, projenin efor ve süresini artıran veya azaltan bir çarpan değeri olarak eklenir.

- COCOMO'nun temel seviyesinde ise doğrudan bir maliyet çarpanı kullanılmaz; ancak eğer projeye özgü bir maliyet hesaplanacaksa, toplam eforu (adam-ay) **birim maliyetle** çarparak projeye özgü maliyet bulunabilir.
- Eğer bir adam-ay maliyeti 100.000 TL olarak belirlenirse, toplam maliyet şöyle hesaplanır:

$$\text{Toplam Maliyet} = 239,87 \text{ adam-ay} * 100.000 \text{ TL} = 20.398.700 \text{ TL}$$

Maliyet etmeni		Seçenekler					
		Çok Düşük	Düşük	Normal	Yüksek	Çok Yüksek	Oldukça Yüksek
Ürün Özellikleri	RELY	0,75	0,88	1,00	1,15	1,40	-
	DATA	-	0,94	1,00	1,08	1,16	-
	CPLX	0,70	0,85	1,00	1,15	1,30	1,65
Bilgisayar Özellikleri	TIME	-	-	1,00	1,11	1,30	1,66
	STOR	-	-	1,00	1,06	1,21	1,56
	VIRT	-	0,87	1,00	1,15	1,30	-
	TURN	-	0,87	1,00	1,07	1,15	-
	ACAP	1,46	1,19	1,00	0,86	0,71	-
	AEXP	1,29	1,13	1,00	0,91	0,82	-
	PCAP	1,42	1,17	1,00	0,86	0,70	-
	VEXP	1,21	1,10	1,00	0,90	-	-
	LEXP	1,14	1,07	1,00	0,95	-	-
	MODP	1,24	1,10	1,00	0,91	0,82	-
Proje Özellikleri	TOOL	1,24	1,10	1,00	0,91	0,83	-
	SCED	1,23	1,08	1,00	1,04	1,10	-

- Rely: Yazılımın güvenilirliği
- Data: Veri Tabanının Büyüklüğü
- Cplx: Karmaşıklığı
- Time: İşletim zamanı kısıtı
- Stor: Ana Bellek Kısıtı
- Virt: Bilgisayar Platform Değişim Olasılığı.
- Turn: Bilgisayar İş Geri Dönüş Zamanı. Hata düzeltme süresi.
- Acap: Deneyim, Birlikte çalışabilirlik.
- Aexp: Proje ekibinin ortalama tecrübesi.
- Pcap: Programcı Yeteneği.
- Vexp: Proje ekibinin geliştirilecek platformu tanıma oranı.
- Lexp: Programlama dili deneyimi
- Modp: Modern Programlama Teknikleri (Yapısal programlama, Görsel programlama, Yeniden kullanılabilirlik)
- Tool: Yazılım Geliştirme araçları kullanımı (CASE araçları, Metin düzenleyiciler)
- Sced: Zaman Kısıtı.

- Maliyet çarpanı (C) 15 maliyet etmeninin çarpımı sonucudur. ($C = C_1 * C_2 * C_3 * \dots * C_{15}$)
- Maliyet çarpanı C ile K değeri çarpılarak düzeltilmiş iş gücü değeri olan Kd değeri elde edilir ($K_d = K * C$).
- Kd değeri temel modelde kullanılan zaman formüllerinde kullanılarak zaman maliyeti hesaplanır.

- **Cocomo Modeli:**

- Kodu temel alır ve proje için gerekli eforu tahmin ederken **kod satırı sayısını (KLOC - Thousand Lines of Code)** kullanır.
- Cocomo, yazılım geliştirme sürecinde harcanacak adam-ay eforunu hesaplamaya odaklanır.
- Genelde kodun karmaşıklığı, personel deneyimi, geliştirme ortamı gibi **faktörleri çarpan olarak ekleyerek** daha detaylı tahmin yapar.

- **İşlev Noktaları:**

- Projeyi yazılımın **işlevsel gereksinimlerine** dayandırır ve kullanıcıya sunulacak işlevlerin sayısı ve karmaşıklığını temel alır.
- Kod satırı sayısını göz önüne almaz; koddan bağımsız bir ölçüm sunarak **geliştirme öncesi tahminleme** yapmayı kolaylaştırır.
- **Kullanıcıya sunulan işlevleri** temel aldığı için yazılımın kullanıcı gereksinimlerine ne kadar uygun olduğunu ölçmeye odaklanır.

- **Cocomo:** Ölçüm birimi olarak **Kod Satırını (KLOC)** kullanır. Projenin toplam kod satırı tahmin edildiğinde efor, maliyet ve zaman tahminlemesi yapılır. Kod satırına dayandığı için uygulama geliştirme öncesinde net bir kod sayısı tahminine ihtiyaç duyar. Bu da proje başlamadan tahmin yapmayı zorlaştırabilir.
- **İşlev Noktaları:** Ölçüm birimi olarak **İşlev Noktaları (FP)** kullanılır. İşlev noktaları, yazılımın işlevsel gereksinimlerini ve karmaşıklığını ölçerek projeyi değerlendirir. Bu yüzden, kod miktarıyla doğrudan ilişkisi yoktur. Kod yazılmadan önce işlevsel gereksinimlere dayanarak tahmin yaptığı için gereksinim belirleme aşamasında faydalıdır.

- **Cocomo:** Üç seviyede (Temel, Ara ve Detaylı) modelleme sunar ve farklı proje türleri için çeşitli çarpanlarla daha detaylı tahminler yapabilir.
- **İşlev Noktaları:** Kullanıcı açısından değerlendirildiği için yazılımın işlevsel kapsamı hakkında detay verir, ancak yazılım geliştirme süreci hakkında ayrıntılı tahmin sağlamaz.

- **Cocomo:** Özellikle kod odaklı projeler için uygundur ve teknik açıdan daha fazla veri sağlar. Büyük ve karmaşık yazılım projelerinde tercih edilir.
- **İşlev Noktaları:** İş süreçlerine dayalı projeler, ERP sistemleri ve işlev odaklı yazılımlar için daha uygun bir yaklaşımdır. Kullanıcı odaklı gereksinimleri önceleyen projelerde tercih edilir.

- **Cocomo** modeli, daha teknik, kod merkezli bir yaklaşım benimser ve detaylı tahminler için geliştirme sürecinin içinde veya sonunda kullanılır.
- **İşlev Noktaları** ise işlevsel gereksinimlere dayalıdır ve yazılımın kullanıcıya sağladığı işlevsel değeri önceden ölçmeye odaklanır.

Kaynakça

- <https://www.sisasoft.com.tr>
- Dr. Fatih Yücalar ve Dr. Emin Borandağ, Yazılım Mühendisliğinde Modern Yaklaşımlar, Nobel Kitap
- <http://www.kriptarium.com/gereksinim.html>
- [https://bidb.itu.edu.tr/sevir-defteri/blog/2013/09/08/entegre-yetenek-olgunluk-modeli-\(cmmi\)](https://bidb.itu.edu.tr/sevir-defteri/blog/2013/09/08/entegre-yetenek-olgunluk-modeli-(cmmi))
- <https://emrealic.wordpress.com/2013/04/03/cmml-nedir/>
- <https://tr.wikipedia.org/wiki/CMMI>
- <https://www.guncelakademi.com/olcme-nedir--n262.html>
- <https://www.arakatman.com>
- Doç. Dr. Recep ERYİĞİT'in ders notları
- <http://www.yazilimcildunyasi.com/2017/01/cocomo-constructive-costing-model.html>