T.C.
MALTEPE ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

BASAMAKLI CMMI MODELİ İLE EXTREME PROGRAMMING METODUNUN DEĞERLENDİRİLMESİ

Emin BORANDAĞ

Yüksek Lisans Tezi

Tez Danışmanı
Prof. Dr. A. Mesut RAZBONYALI

İSTANBUL – 2006

T.C.
MALTEPE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

BASAMAKLI CMMI MODELİ İLE EXTREME PROGRAMMING METODUNUN DEĞERLENDİRİLMESİ

YÜKSEK LİSANS TEZİ

Emin BORANDAĞ

Tez Danışmanı
Prof. Dr. A. Mesut RAZBONYALI

İSTANBUL – 2006

Bu tez çalışması, Maltepe Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..... / ..... / ..... tarih ve ..... / ..... sayılı kararıyla oluşturulan jüri tarafından ***Bilgisayar Mühendisliği Yüksek Lisansı Tezi*** olarak kabul edilmiştir.

JÜRİ

Prof. Dr. A. Mesut RAZBONYALI
Danışman

Prof. Dr. İlhami YAVUZ
Üye

Prof. Dr. Fuat İNCE
Üye

ÖZET

Yüksek Lisans Tezi, BASAMAKLI CMMI MODELİ İLE EXTREME PROGRAMMING METODUNUN DEĞERLENDİRİLMESİ, T.C. Maltepe Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı.

Bu çalışmada, bir yazılım süreç değerlendirme modeli olan CMMI kullanılarak bir yazılım geliştirme metodu olan Extreme Programming değerlendirilmiştir. CMMI isterlerinin Extreme Programming tarafından karşılanabileceğinin mümkün olup olunmadığıı sorusuna cevab aranmaktadır.

Karşılaştırılma sistemi; CMMI 2.seviye isterleri temel alınarak gerçekleştirilmiştir. CMMI 2.seviye isterleri ele alınıp, Extreme Programming pratikleri ile karşılaştırılmıştır. Sonuç kısmında ise; CMMI isterlerinin Extreme Programming paratikleri tarafından nasıl gerçekleştirildiğinin üzerinde durulmuştur.

Bu tez 2006 yılında yapılmıştır ve 157 sayfadan oluşmaktadır.

Anahtar Kelimeler: *CMMI, XP, Yazılım Kalitesi, Yazılım Güvenliği.*

ABSTRACT

Master Thesis, EVALUATION OF STAGED MODEL CMMI WITH EXTREME PROGRAMMING METHOD, T.C. Maltepe University, Graduate School of Natural And Applied Sciences, Department of Computer Engineering

In this work, the software process assessment model CMMI is used to asses the software development method Extreme Programming. In particular an answer is sought as to what extent Extreme Programming fulfilled CMMI requirements for different maturity levels.

After comparing the CMMI requirement with Extreme Programming practices, it is concluded that Extreme Programming practices can meet maturity level 2 requirements with little difficulty. It is also possible to augment Extreme Programming practices with some organization effort can meet level 3 requirements. But level 4 and level 5 are well beyond Extreme Programming.

This thesis written in 2006 and consists of 157 pages.

KEYWORDS: CMMI, XP, Software Quality, Quality Assurance

TEŞEKKÜR

Yüksek lisans tezimi hazırlamamda bana yardım eden, zaman ayıran ve her koşulda bana destek olan tez danışmanım, T.C. Maltepe Üniversitesi Rektörü hocam Prof. Dr. A. Mesut RAZBONYALI'ya, tez hazırlamamdaki yardımlarından, ilgisinden, alakasından, sonsuz hoş görüsünden ve desteğinden dolayı, T.C. Marmara Üniversitesi Bilgisayar Mühendisliği Bölüm Başkanı Prof. Dr. Fuat İNCE hocama ve T.C. Maltepe Üniversitesi Rektör Yardımcısı ve Fen Bilimleri Enstitüsü Müdürü Prof. Dr. İlhami YAVUZ hocama, çalışma arkadaşlarıma ve aileme sonsuz teşekkürlerimi sunarım.

İÇİNDEKİLER

TABLOLAR

ŞEKİLLER

KISALTMALAR

PM-CMM	People Management Capabilitiy Maturity Model,
RAD	Rapid Aplication Development,
UML	Unified Modeling Language,
XP	Extreme Programming,
ISO	Internatrional Standardizasyon Organization,
AQAP	Allied Nato Quality Assrance Program,
SWEBOK	Software Engineering Body of Knowledge,
KA	Knowledge Areas,
DOD	Department of Defense,
IEC	International Electrotechnical Commission,
JTC	Joint Technical Commite,
CMM	Capability Maturity Model,
SEI	Software Engineering Institute,
SPICE	Software Process Improvment and Capability dEtermination,
TQM	Total Quality Management,
MSG	Management Steering Group,
SEPG	System Engineer Process Group,
TWG	Team Working Group,
CMMI	Capability Maturity Model Integration
RFP	Request for Proposal

Giriş

Yazılımda kalite ve yazılım metodolojilerine bakmadan önce temel tanımlara bir göz atacak olursak. Bu temel tanımlardan biri olan yazılım mühendisliğine bakmamız gerekir. Yazılım mühendisliği, yazılım üretiminin mühendislik yöntemleriyle yapılmasını öngören ve bu yönde yöntem, araç, teknik ve metodolojiler üreten bir disiplindir. Bu bakımdan yazılım mühendisliği bir yöntemler kümesi, teknikler kümesi ya da araçlar kümesi olarak değerlendirilebilir. Yazılım mühendisliğinde, yazılım yaşam döngüsünde belirtilen aşamaların gerçekleştirilmesi yazılım üretimi için çok önemlidir. Yazılım ise en basit tanımı ile bilgisayar sisteminde donanım dışında kalan her şey olarak da tanımlana bilir. Yazılım; bilgi, yöntem, mantık ve isterlerin bir araya gelmesi ile oluşturulabilir.

Yazılım mühendisliği ve yazılım metot ve modellerinin tarihine bakmak için 80 yıllık mazisi olan kalite kontrol çalışmalarına bakmamız gerekir. İlk olarak 1924 yılında yapılan kalite kontrol çabalarıyla başlayan bu süreçte yapılan işlerin daha kısa sürede daha az maliyetle nasıl üretileceği sorusu üzerinde durulmuştur. 1940 yılında gelindiğinde modern istatiksel kalite kullanımının başladığı görülür. 1946 yılına ise ilk hafızalanmış bilgisayar olan ENIAC bilgisayarı yapılmıştır. Yapılan bu ve buna benzer bilgisayarlar ile birlikte bilgisayarlarda kullanılacak bilgisayar dili üretme ihtiyacı ortaya çıkmış ve 1953 yılında fortran bilgisayar dili üretilmiştir. Üretilen bu diller ile birlikte ihtiyaç duyulan yazılımların gerçekleştirilmesi sağlanmıştır. Bu bilgisayar dilleri sayesinde üretilen yazılımlar daha kolay bir şekilde üretilse de yazılımlar dan beklenilenler gün geçtikçe artmıştır. Yazılımlardan daha fazla istere cevap verme gereği ortaya çıkmıştır.

1960 yılına gelindiğinde ise kalite kontrol ve yönetim işinin önemi anlaşılmış bazı ülkeler tarafından ulusların en önemli meselesi olarak ele alınmıştır. 1962 yılında meydana gelen NASA'nın bir uzay mekiğini kaybetmesinin nedeninin yazılımdan kaynaklanan bir hatadan olduğunun saptanmasından sonra yazılım üretme belli modellere bağlanması gereği ortaya konmuştur. 1968 yılına gelindiğinde ise ilk kez yazılım mühendisliği terimi kullanılmıştır. Üretilen

yazılımların büyük bir kısmının verimli olmaması, istenilenleri tam olarak karşılamaması ve yazılımların çöpe gittiğinin görülmesi sonucu "yazılım krizi" terimi 1968 yılında kullanılmıştır. 1976 yılında ise yazılım üretme modelleri geliştirilmiştir. 1980 yılında ise toplam kalite çalışmalarına verilen önemi artmıştır.

Yazılım krizi sonrasında geliştirilen yazılım modellerinden biri olan ve 1990'lı yıllarda geliştirilen CMM, zaman içerisinde yapısını ve adını değiştirerek daha güçlü bir şekil alarak adını CMMI olarak değiştirmiştir. CMMI, yazılım şirketlerinin kaliteli yazılımlar üretmesini ilke edindi. Amerikan ordusu için geliştirilen CMMI zaman içerisinde yazılım geliştirme sektöründe de kullanıldı.

Yazılımların büyük bir kısmı geliştirilen bu modeller ve metotlarla oluşturuldu. Gerçekleştirilen yazılımların başarı oranları arttı ama hala bazı önemsenmesi gereken sorunlar vardı. Bu sorunlar ise yazılımda; sürümler geç çıkıyor, hatalar geç fark ediliyor, zaman içerisinde gelen isterlere göre sistem kendi yapısını geliştiremiyor, değişim maliyeti yüksek gibi eleştiriler ortaya çıktı. Bu sorunları aşılması yönünde çalışmalara yapılması sonucu "agile" denilen esnek yöntemler adı verilen yazılım yaşam döngüsünü baştan aşağı değiştiren yöntemler geliştirildi. Bu yöntemlerden en öne çıkanı ise XP oldu.

Genel tanımları ve tezin temelini oluşturan metot ve modelleri kısaca bahsettikten sonra tezin amacı olan; yukarıda bahsedilen esnek modellerden biri olan XP ile CMMI'in bir arada değerlendirilip değerlendirilmeyeceğini ortaya koyarak.CMMI seviye 2'nin istediklerini XP'nin yaşam döngüsü ve pratikleri tarafından yapılıp yapılmayacağına bakmaktır. Pek çok akademisyen ve bilim adamının katkısı ile geliştirilen CMMI ile "agile" tekniklerden biri olan XP ile ilgili pek çok ayrı çalışma yapılmış ve genel itibari ile iki yöntem arasındaki farklılıkların, avantajların ve dezavantajların neler olduğu saptanmıştır. Bu iki yöntemin bir arada kullanılması ile ilgili olarak bazı ufak çaplı çalışmalar yapıldı ise de bu işin derinlemesine inilip bu iki modelin hangi avantajlarının bir arada kullanılmasının nasıl daha efektif bir yöntem çıkarılacağı sorusu üzerinde yeteri kadar durulmamıştır. Bu tez ile birlikte bu iki yöntemin nasıl bir arada kullanılacağı ve

CMMI'ın isterlerini XP'nin nasıl sağlayacağı sorusuna cevap aramaktadır. Özellikle Türkiye gibi yazılım üreten şirketlerin orta büyüklükte olduğu bir ülkede CMMI ile XP kullanımının bir arada yapılmasının ne gibi avantajlar sağlayacağı da tezin sonuçlar kısmında anlatılmıştır.

Tezin kapsamına bakacak olursak; birinci kısımda yazılım ve yazılım mühendisliği konusu üzerinde durularak yazılım karmaşıklığı, yazılım hataları, yazılım yaşam döngüsü ve metodolojilerden örnekler verilmiştir. İkinci kısmında ise yazılımda kalite kavramı ve süreç modelleri hakkında bilgiler verilmiştir. Üçüncü kısımda ise yazılım metodolojileri hakkında detaylı bilgiler verilmiştir. Dördüncü kısımda CMMI detaylandırılarak süreçleri ve seviyeleri anlatılmış. Beşinci bölümde ise XP hakkında son derece derinlemesine bilgiler verilmeye çalışılmıştır. Son bölüm olan altıncı kısımda ise CMMI seviye 2'nin isteklerini XP tarafından nasıl karşılanacağı üzerinde durulmuş ve tez sonunda elde edilen sonuçlara yer verilmiştir.

1.Yazılım ve Yazılım Mühendisliği

1.1Yazılım Mühendisliği

Yazılım Mühendisliği, yazılım üretiminin mühendislik yöntemleriyle yapılmasını öngören ve bu yönde yöntem, araç, teknik ve metodolojiler üreten bir disiplindir. Bu bakımdan yazılım mühendisliği bir yöntemler kümesi, teknikler kümesi ya da araçlar kümesi olarak değerlendirilebilir. Yazılım üretiminde, yazılım yaşam döngüsünde belirtilen aşamaların gerçekleştirilmesi yazılım mühendisliği için ön koşuldur.[1]

Yazılım mühendisliğinde ortaya çıkarılacak ürünlerin sınırlarının, fiziksel sınırlardan çok zihinsel düşünce sınırları olduğunu da belirtmek gerekmektedir.

Yazılım mühendisliği, yazılım üretimindeki karmaşıklıkları gidermeyi hedefler. Geçmişte kullanılan iş akış şemaları gibi yöntemler, günümüzde, yazılım projelerinin boyutlarının büyümesinden dolayı yetersiz yerlerini daha gelişmiş metotlara bırakmıştır. Bundan dolayı artık yazılım üretim işi, tek kişinin başarabileceği boyuttan çıkmış ve takım işi biçimine dönüşmüştür.

Yazılım mühendisliği bu temel olgular üzerine önem kazanmıştır. Ayrıca yazılım mühendisliği sonunda çıkan ürünün, elle tutulamayan kavramsal bir ürün olması nedeni ile diğer mühendislik ürünlerinden farklıdır. Yazılımların ortaya koyduğu ürünün maddi değerinin yüksekliği ve bu yüksek değeri ortaya koymak için gereken ihtiyaçların çok küçük olduğunu da göz önüne alırsak devlet politikası içerisinde yazılım mühendisliğine ayrıca önem verilmesi gerekmektedir. Devlet tarafından ayrılan bu kaynakların geri dönüşü, doğru bir planlama ile kısa bir süre içerisinde gerçekleşecektir. Hem bu sayede cari açığını azaltacak yönde bir yarar sağlayacaktır. [2]

Yazılım mühendisliğinin teknoloji ile ilişkisine bakarsak; teknolojideki gelişmelerin altında, bilgisayar teknolojisinin olduğu görülecektir. Bu teknolojiyi

kullanmamızı sağlayan şey ise geliştirilen yazılımlardır. Günümüzde çok yüksek kalitede yazılımlar, üretmek mümkündür.[3]

Yazılım mühendisliğinin eğitimle olan ilişkisine bakacak olursak, yüksek lisans ve doktora düzeyinde verilen yazılım mühendisliği eğitimi şimdilerde lisans düzeyinde de verilmeye başlandığı görülecektir. Günümüzde üniversiteler, yazılım mühendisi bölümünde kişileri eğitmekte ve bu daldan eğitim alan kişilere bilgisayar mühendisliği yerine "yazılım mühendisliği" diploması vermektedir.[4]

Bir sonraki bölümde yazılım ile ilgili bilgiler verilerek yazılım ve yazılım mühendisliği ile ilgili konular pekiştirilecektir.

1.2Yazılım

Müşterinin ihtiyaçları doğrultusunda verilen isterlerle birlikte yazılımcının kendi bilgisini kullanarak, yazılım araçları ve yazılım teknikleri ile oluşturulan sistemin ihtiyaçlarını karşılayan çalışan kodların bütününe yazılım denir.

Bir sistemin donanımı dışında kalan her şey olarak da tanımlanan yazılım çeşitli bilişimlerden oluşur.[5]
Yazılım bileşenleri:

Yazılım = Bilgi + Yöntem + Yazılımcı + Mantık + İsterler [6]

1.3 Yazılımın Sınıflandırılması

Tek bir yazılım çeşidi ya da tek bir yazılım metodunu öğrenerek bütün yazılım çeşitleriyle ilgili bilgileri aktarmak mümkün değildir.Veri tabanı ile ilgili yazılımlar olsun, grafiğin ön planda olduğu animasyon ile ilgili yazılımlar olsun hepsinin farklı bir yapısı ve işleyişi vardır. Aşağıda farklı yazılım çeşitlerinden örnekler verilmiştir.[7]

- Animasyon Yazılımları
- Mesleki ve Ticari Yazlımlar
- Sistem Yazılımları
- Operating System
- Compiler
- Editörler
- Debug programları
- Yapay Zeka Yazılımları

1.4 Yazılım Mühendisliğinin Tarihi

Yazılım mühendisliğinin tarihine baktığımızda ilk hafızalanmış bilgisayar programına bir göz atmak gerekir. İlk hafızalanmış bilgisayar programı, 1946 yılında ENIAC bilgisayarda kullanılmıştır. Başlangıçta bilgisayarla ilgili sorunlar sadece hata olarak adlandırıldı. Daha sonraları bilgisayar hataları, “bug” adını aldı hata düzeltmeye ise “debug” adı verildi.[8]

1953 yılına gelindiğinde yazılım işinin kalbi İngiltere den ABD’ ye taşındı ve çeşitli bilgisayar dilleri üretildi. Bu dillerden birisi de 1953 yılında üretilen fortran bilgisayar dili idi. Bu arada bilgisayar dilleri kullanılarak pek çok yazılım geliştirildi. Ancak geliştirilen bu yazılımları üretmek için kullanılan metotlar önceleri yeteri kadar önemsenmedi. 1962 yılında NASA tarafından uzaya gönderilen bir aracın infilak etmesinin nedeninin yazılımlardan kaynaklanan bir hata olduğu öğrenilince yazılım hataları üzerinde daha önemle ve daha özenle durulmaya

başlandı. 1968 yılına gelindiğinde Almanya'da düzenlenen bir konferansta yazılım mühendisliği kavramı ilk kez kullanıldı. Bu kavramın tanımlanması ile birlikte yazılım işi, bir disiplin içerisinde değerlendirmeye başlandı. Çünkü üretilen yazılımların büyük bir kısmı, istenilenleri karşılamıyor, çöpe atılıyor ya da kullanılmıyordu.

1972 yılında ise IEEE tarafından yazılım mühendisliği kavramı kullanıldı. 1976 yılında gelindiğinde ise yazılım geliştirme standartları tanımlandı. 1990'lı yıllardan itibaren bilgisayar mühendisliği disiplini içerisinde yer alan yazılım mühendisliği, daha sonraki yıllarda diğer mühendislik dallarından bağımsız bir hale geldi. Yazılım mühendisliği, teknolojinin gelişimine büyük bir ivme sağlamıştır. Yaşamımız boyunca kullandığımız pek çok araç gerecin içerisinde yazılım teknolojisini görmek mümkün olmaktadır. Aynı şekilde yazılım mühendisliği, değişik sektörlerin (bankacılık, sağlık, telekomünikasyon, hava sanayi, eğitim, finans, vb) işlerini daha kolay bir şekilde yapmaları için yardımcı olmaktadır. [9]

1.5 Yazılımın Tarihsel Aşamaları

Yazılımın gelişimindeki tarihsel aşamaları dört grupta toplayabiliriz:

- **Birinci Dönem:** İlk bilgisayarların üretilmeye başladığı bu yıllarda bilgisayarların maliyetlerinin yüksekliğinden dolayı kullanıcılar bire bir olarak bilgisayar kullanamadıktaydı.[10]
- **İkinci Dönem:** Veritabanı kullanımında başladığı bu dönemde, bilgisayarların maliyet sorunları da belli oranlarda azalmıştır.
- **Üçüncü Dönem:** Bilgisayar ağ teknolojilerinin geliştiği bu dönemde, kişisel bilgisayarlarda kullanılmaya başlanmıştır.
- **Dördüncü Dönem:** Şu an içinde bulunduğumuz yılları da kapsayan bu dönemde bir işi yaptırmak için birden fazla bilgisayar kullanılma teknolojileri ortaya çıkmıştır. Yazılım üretiminde karşılaşılan zorluklar nedeniyle yazılımda kalite anlaşışı önem kazanmaya başlamıştır. Bu amaçla çeşitli kuruluşlar kendi yazılım standartlarını geliştirmişlerdir.

1.6 Hangi Şartların Oluşması ile Yazılım Üretilmesine İhtiyaç Vardır?

- Belli bir yapıya oluşturulması gereken bir sistem (Bilgisayar teknolojisi kullanmak isteyen bir sistem),
- Çok sık değişen personel (Yapıyı bilmeyen personel),
- Bir bilgisayar sistemi ve yazılım olmadığı için yeni gelen personele işlerin anlatılması için geçen süre ve bu uzun süre içerinde artan maliyet,
- İş süreçlerinin uzaması,
- Standart ve yöntem eksiklikleri,
- İş yeri kaynaklarının yeteri kadar iyi değerlendirilememesi,
- Varsa mevcut yazılımın zaman süresi boyunca değişen ihtiyaçları karşılayamaması,
- İş oluşturma süresi ile paralel olarak artan maliyet sorunu. [11]

1.6.1 Yazılım üretmek için gerekenler

- Yazılım ihtiyacı,
- Yazılım üretmek için ayrılacak kaynak,
- Değişik yeteneklere sahip personel (belgeleme uzmanı, sınayıcı programcı, çözümleyici),
- Yazılım çıktısı ile ilgilenen ve bilgi teknolojileri konusunda ilgili olmayan kullanıcılar,
- İhtiyaçların karşılanıp karşılanmadığıyla ilgilenecek yönetici,
- Yazılımı kullanacak kişi,[12]

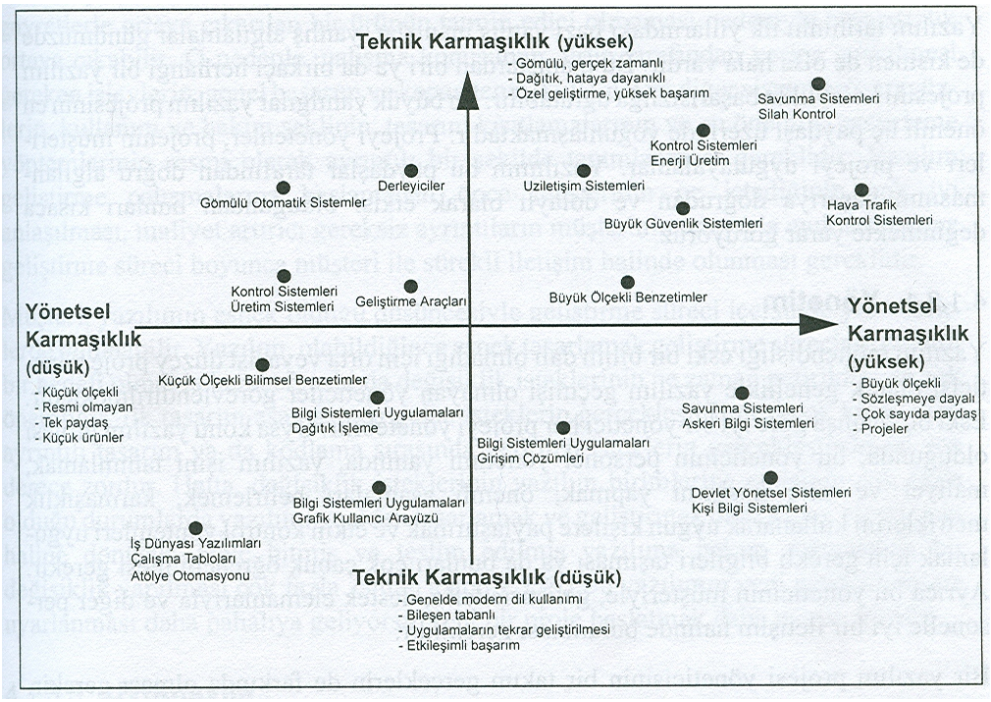
1.6.2 Yazılım geliştirme sıkıntıları

- Yeterli bilgiye sahip olmadıklarından ne istediğini tam olarak anlatamayan kullanıcılar.
- Değişikliklere reaksiyon gösteren yöneticiler.
- Yeterince tanımlanmamış kullanıcı beklentileri.
- İyi olarak anlaşılamayan isterler sonucu yanlış işlerin yapılması.
- Yeteri kadar yazılımların test edilememesi.
- Zamanlama, proje kestirim süresi yanılgıları.

1.7 Yazılım Karmaşıklığı

Bilgisayar sistemlerinin zaman içerisinde yapılarındaki gelişme sistemlerin karmaşık bir yapıya bürünmelerini sağlamıştır. Taleplerin artması ile birlikte daha işe özel ve daha komplike yazılımlar çıkmıştır. Ortaya çıkan gereksinimleri karşılamak giderek güçleşmiş ve yöntemsiz bir şekilde işin içinden çıkılması imkansız bir hal almıştır.

Bilgisayar tabanlı sistemlerin bu günkü kullandıkları yazılımları, karmaşıklıkları bakımından bir tablo üzerinde göstermek mümkündür.[13]



Şekil1.1 Yazılım Karmaşası

Yukarıdaki şekildeki yazılım karmaşıklığı, iki farklı boyuta ele alınmıştır. Bunlardan birincisi yönetimsel karmaşıklık ikincisi ise teknik karmaşıklıktır. Burada üretilecek yazılımın türüne göre bir karmaşıklık hesabı kabaca yapılabilir. Yazılım karmaşıklığının en önemli nedeni kod sayısındaki artıştır. Bu kod sayısındaki artış yazılımda hata oluşma riskini arttırır. Zaman içerisinde artan isterlerle birlikte daha da kendini gösteren kod sayısındaki artış, yazılımda ortaya çıkması muhtemel hataların sayısını arttırmaktadır. Bu yazılım hatalarını mümkün olduğu kadar azaltmak için bütünleşme, test ve bakım kısımlarına özen gösterilmesi gerekmektedir.

1.8 Yazılım Hataları

Yazılım sırasında oluşan hatalar yazılımın gerçekleşme süresini uzatmanın yanında yazılım gelişimi sırasında bulundukları yere göre maliyetleri de arttırır.

Kullanılan yazılımlardaki tüm hataların bulunması oldukça zordur. Bu nedenle yazılımların içerisindeki bütün hataların bulunup, ondan sonra sisteme geçilmesi gibi bir konu söz konusu değildir. Yapılan testlere göre bulunan hatalar çözümlenir.

Aşağıdaki tabloda oluşan hatların dağılımları verilmiştir

Tablo1.1 Yazılım Hatalarının Dağılımı

Yazılım Hatalarının Dağılımı	
Tasarım Sırasında Oluşan Hatalar (Mantıksal Tasarım)	% 20
İşlevsel Tasarım	% 15
Kodlama	% 30
Belgeleme	% 35

Yazılım oluşturulması sırasında ortaya çıkan hatalar ileriki safhalarda ortaya çıkıyorsa maliyeti erken safhada çıkan hatalara göre daha da fazladır. Aşağıda yazılım üretme safhalarına göre oluşan hataların maliyetleri verilmiştir.

Tablo 1.2 Hata Düzeltme Maliyetleri

Hata Düzeltme Maliyetleri	
Çözümleme	1
Tasarım	5
Kodlama	10
Test	25
Kabul	50
İşletim	100

1.9. Yazılım Proje Yönetimi ve Yazılım Mühendisliği

Yazılım ve yazılım mühendisliği kadar önemli onan bir diğer konu da projenin iyi bir şekilde yönetilmesidir. Kendi yapısını koruyan sağlam bir yazılımın proje kısmı insan ve süreç konuları üzerine odaklanması gerekmektedir.

Yazılım proje yönetiminin ilgilenmesi gereken birinci kısım insandır. Yazılım üretilmesi aşamasında çok yoğun olarak kullanılan insan faktörü proje yöneticileri tarafından ihmal edilmemesi gereken bir konudur. Hem yazılımı üretecek kişiler hem de yazılımı isteyen kişiler ile yazılım proje yöneticilerinin sürekli bir araya gelerek proje kapsamı ve şu an gelinen durumu konuşmaları gerekmektedir. Eğer yönetici projenin erken safhasında yeteri kadar müşteri ile bir araya gelmez ise ileriki safhalarda çözüm üretememe riski ile yüzleşmek zorunda kalabilir. Sorunlar işin ikinci kısmıdır. Sorunlar ile ilgili risk değerlendirmeleri sırasında felaket senaryoları oluşturulmalıdır. Erken sorunun tespiti yazılım yöneticisinin hatayı daha kısa sürede önlemesini sağlayacaktır. Süreç kısmında ise süreçler yeteri kadar belgelenmeli süreçlerin girdileri ve süreç sonucunda çıkacak bilgiler irdelenmelidir. Proje yöneticisi işlerini yapabilmek için uygulamadan gelen bir deneyime, sorunlar için kullanılacak teknik araçları ve yöntemlere sahip olmalıdır. Bu işleri yeteri kadar iyi bir şekilde yapılmaz ise sorunlara yanlış çözümler üretilmesi gibi bir konu ile karşılaşabilir. [14]

1.9.1 Yazılım ve insan ilişkisi

Yazılım üretme aşamasında son derece önemli olan insan olgusu, çeşitli yazılım geliştirme modellerinde üzerinde durulan bir model olmuştur. Hızlı bir şekilde yeni işleri gerçekleştirebilecek yeteneğe sahip insan seçilmesi 1960'ların başından beri tartışılan bir konudur. Aslında insan faktörü, yazılım geliştirmede o kadar önemlidir ki yazılım üretilmesi ve organizasyonlarının başarısı yüksek oranda seçilecek personelin başarısıyla doğru orantılıdır. Uygulamaların kalitesiyle seçilen insanlar arasındaki ilişkiyi PM-CMM, yazılım üreten yazılım mühendisleri için eleman seçimi, ekibin bir araya getirilmesi, performans sağlanması sağlanan performansın yönetimi, personelin belli sürelerdeki eğitimi, yaptıkları ile ilişkili olarak şirketler içindeki kariyer gelişimi, süreçlerin ve iş akışlarının planlanması ve takım ile kültür gelişimi alanlarını tanımlar.[15]

1.10 Yazılım Mühendisliği Eğitiminin Bireye Kazandırdıkları

Yazılım mühendisliği eğitimi doğru kurumlardan doğru şartlarda alındığı taktirde şu sonuçlara ulaşılacaktır:

- Yazılımı kullanacak kişilerden gelen isterleri analiz ederek, bu isterlere uyan çözümler tasarlayabilmek,
- Yazılımı kullanacak kişilerin belirlediği proje bitirme tarihi, proje bütçesi, yazılım kullanılabilirlik konularında bir işbirliği sağlaya bilmek
- Mühendislik etiğine uygun yasalar çerçevesinde belirlenmiş kurallara dikkat eden ve aynı zamanda müşterinin beklentilerini ekonomik bir çözüm üretebilmek
- Yazılımın yaşam döngüsü içerisindeki bütün aşamalarını gerçekleştirerek yazılım teorileri, modelleri ve tekniklerini özümsemek ve uygulayabilmek. [16]

1.11 Yazılım Temel Adımları

Yazılımın oluşması sırasındaki aşamaları ve yazılımın kullanım aşamalarını kapsamaktadır. Yazılım yaşam döngüsü temel olarak beş aşamadan oluşur.

- **Planlama:** Üretilecek yazılımın bütün ihtiyaçlarını saptandığı kısımdır. Bu aşamada personel ihtiyaçları, donanım gereksinimleri, verilen işin yapılabilirliği sınanır.
- **Çözümleme:** Yazılım işlevleriyle gereksinimlerin çıkarıldığı kısımdır. Burada mevcut sistem incelenir sistemde yapılması gerekenler ortaya konur istekler belirlenir.
- **Tasarım:** Geliştirilecek bir modelin ilk ürününü ortaya çıkartmaktır. Mevcut sistemden farklı olarak ilk kez ortaya çıkarılacak büyük yapısal değişiklikler belirtilir.
- **Gerçekleştirim:** Gerçekleşen tasarımın kodlanmaya başlandığı kısımdır. Bu aşamada sınama isleride gerçekleşir.
- **Bakım:** Hata giderme ve yeni gelen isterlerin gerçekleştiği kısımdır.[17]

1.12Yazılım Yaşam Döngüsü

Yazılım projelerinde projenin yapısına dikkat edilerek kendi yapısına uygun bir yaşam döngüsü kullanması sistemlerin gelişmesi bakımından uygun olmaktadır. Yazılım yaşam döngüsünde her proje, sistem analiz, tasarım, kodlama ve test aşamalarından geçer.

Dikkat edilmesi gereken kurallar:

- Kişisel isteklere göre yazılımı oluşturmaktan ziyade belli kriterlere dayandırılarak yazılımın oluşturulması daha önemli olacaktır. Bu sayede personel değişiklikleri yaşansa bile yazılım üretim ortamı bu değişikliklerden daha az etkilenecektir.
- Yazılım yaşam döngüsü olmadığı takdirde, geliştirilen yazılımların proje içerisinde belirlenen noktalara belirlenen tarihte geldiğinin denetlenmesi ve gelişimlerinin izlenmesi güçleşecektir.

1.13Yazılım Ürünleri

- Çalışan bir yazılım
- Belgeler
- Eğitim
- Yazılım Parçaları

1.14 Yazılım Projelerinde Başarı

Yazılım projelerinde başarı ilkeleri

- İstekleri yerine getirme
- Zamanında bitirme
- Bütçesi içerisinde bitirme

Yazılım mühendisliğin ilk çıktığı 1960'lı yıllarda ve yazılımda kalite sorusunun tartışılmaya başlandığı 1970'lı yıllara kadar üretilen projelerin başarı oranları:

Tablo 1.3 Proje Başarısı Oranları

Tam Başarı	Kısmen Başarı	Çöpe Gidenler
%4-5	%45-50	%45-50

Görüldüğü gibi firmaların çoğu belli metot ve metodolojileri kullanmadığı veya tam olarak bu yöntemlere başvurmadığı için proje geliştirmeye çalıştıklarında büyük sorunlarla karşılaşmışlardır. Bunun sonucu olarak ta ilk kez 1968'de söz edilen "Yazılım Krizi" sözcüğü çok sık kullanılır olmuştur. Bu sorunun farkına varılmış ve 1984 yılında Software Enginering Institue(Carengie Mallon Üniversitesi ve IBM) tarafından oluşturulan bir çalışma ekibi ile o güne kadar oluşturulmuş projeleri incelenmiştir. Bu ekip başarılı olan projeleri inceleyerek bu projelerin çoğunda yer alan ortak değerleri hazırladıkları raporda "Best Practices" adı ile dile getirmişlerdir.

"Best Practices"

- Metot Kullan
- Soruları Erken Bul Erken Çöz
- Sorunların Erken Nedenlerini Bul
- Müşteri Memnuniyetini Ölç
- Preje Kestirim Aracı Kullan
- Süreçleri İyi Belirle(process)
- Kod Denetlemesi Yap
- Konfigürasyon Yönetimini Araçlar Kullanarak Yap
- Ürün Değil Süreç Önemli
- Süreçlerin Eğitimini Ver

1.15Yazılımda Bulunması Gereken Temel Özellikler

Her yazılımda olmazsa olmaz bazı özellikler vardır, aşağıda bu özelliklerden bazıları verilmiştir.

- **Bakım Yapılabilir Olmalı:**Gerçekleştirilecek yazılımın sürekli geliştirilebilir, düzetme yapılabilir, yeni işlevler eklenebilir olması gereklidir.
- **Güvenilir:**Yazılım her koşul altında aynı sonuçları vermeli, herhangi bir çökme olayı olmadan varlığını sürdürerek çalışa bilmeli.
- **Verimli:**Kaynakları(Zaman, bellek, disk, işlemci) en iyi şekilde kullanmalı, en az isterle elde edilebilecek en iyi sonucu almalı.
- **Kullanışlı:**Kolay anlaşılabilir, müşteri tarafında kullanıcı dostu, kullanıcı analizi yapılmış, yetkilendirme sağlanmış ve iyi belgelenmiş olmalı.

1.16 Yazılım Geliştirme Metodolojileri

Yazılım geliştirmek için bazı metot ve metodolojiler kullanmamız gerekmektedir. Metot ve metodoloji kullanmak "Best Practices" lerden biridir. Metod ve metodolojiler yapılan işleri belli bir proje yapısı içersinde süreçlere ayırarak neyin ne şekil de yapılması gerektiğini bize gösterir. Aynı zamanda yapıyı düzgün bir biçimde gerçekleştirmemizi sağlar. Aşağıda kullanılan bazı yazılım geliştirme metodolojileri yer almaktadır.

1-Çağlayan Modeli
2-Evrimsel Model
3-Prototipleme
4-RAD(Rapid Aplication Development)Hızlı Uygulama Geliştirme
5-Spiral Model (Barry Boelim)
6-Montaj (Parçalardan oluşturma)
7-Formal Teknikler (Matamatiksel)
8-)UML (Unified Modeling Language)
9-)Çevik (AGİLE)Metotlar (eXtreme Programming)
10-)Concurrent (Paralel)Geliştirme Metodu
11-)4.Kuşak Diller

12-)Yapısal Metotlar

13-)Nesne Tabanlı (Yönetimli)(OO,Object Oriented)

14-)Michael Jackson Metodu

15-)Süreç Yaklaşım

1.16.1 Metodolojiler

Günümüzde yüzden fazla metodoloji geliştirilmiştir. Geliştirilen bu metodolojilerde genelde çağlayan ya da Helezonik (Spiral) modeli temel alınmıştır. Diğer bir taraftan özel kuruluşlar da uzmanlar tarafından geliştirilmiş ve birçok kuruluş tarafından benimsenmiş ve uygulanmakta olan çeşitli metodolojileri vardır. Bir metodolojide bulunması gereken temel bileşenler ya da özellikler:

- Ayrıntılı bir süreç modeli
- Ayrıntılı süreç tanımları
- İyi tanımlı üretim yöntemleri
- Süreçler arası 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 ara yüz ve ilişki modeli
- Standartlar

Bir kuruluşun kendi metodolojisini geliştirmesi ve uygulaması oldukça zaman alıcı ve uzmanlık gerektiren bir konudur. Tam kapsamlı bir metodolojinin, konu uzmanları tarafından geliştirilmesi için en az 50 kişilik bir gurubun bir aylık çalışması gerekmektedir.[18]

1.17 Yaşam Çevrimi

İsterlerin Alınması ve Planlama:

Proje gerçekleştirilmeye başlamadan her projede olduğu gibi, müşterinin isteklerinin proje sonunda karşılanıp karşılanmayacağına bakılır. Yani müşteriden gelen isterlerin makul isterler olup olmadığına bakılır. İsterleri alırken buradaki isterlerin hangi kullanıcıdan alınacağı öncelikli olarak belirlenir. Bu aşamadan sonra mevcut sistem incelenir, mevcut sistemdeki hatalar eksiklikler ortaya koyulur. Yeni oluşturulacak sistemin amaçları ve hedefleri belirlenir. Oluşturulacak sistem sonucunda hangi getirilerin oluşacağı belirlenir.

Sistem İncelenmesi:

Tam anlamıyla sistemin bir analizi yapılır, burada sistem çözümlemek için çeşitli yöntemler kullanılır. Burada kullanıcının yazılım bittikten sonra yapacakları işler modellenir. Kullanıcı istekleri detaylı olarak gözden geçirilir. Bu işlerden sonra maliyet ortaya konur ve bir iş akışı programı yapılır.

Sistem Tasarımı:

Sistem tasarımı proje için önce genel bir tasarım gerçekleştirilmesiyle başlar. Daha sonrasında bu genel tasarım yapısını bozmadan projenin alt kısımları için daha detaylı bir tasarım hazırlanır. Tasarımları hazırlamak için eldeki imkânlar değerlendirilir. Sahip olunan bilgi birikimine bakılır. Bu tasarımlar oluşturulurken sadece yazılımcı tarafındaki kişilerle birlikte müşteri temsilcilerinin de olması projenin daha sağlam temleler üzerine oturtulmasına yardımcı olacaktır.

Sistem Gerçekleştirimi:

Sistem incelenmesi ve sistem tasarımından gelen bilgi birikimi sistem gerçekleştirimine bir alt yapı oluşturur. Sağlanan bu alt yapı ile sistem gerçekleştirilmesi işlemine geçilir. Burada belli kurallara göre belli bilgisayar dillerinde yazılan kaynak kodlarıyla sistem gerçekleştirilmesi sağlanır.

Sistem Testi:

Yazılımın gerçekten işe yarayıp yaramadığı yapılan işlerin doğru olup olmadığıyla ilgilidir. Yapılan yazılımın da doğru çalışıp çalışmadığı ancak onun test edilmesiyle anlaşılır. Sistem testi sırasında gerçekleşen testler sistem gerçekleştirme aşamasından sonra da devam eder. Bu sayede sistemdeki aksaklıklar ve yanlışlıklar ortadan kaldırılır.

Sistem Bakımı:

Yazılım test aşamasından sonra müşteriye verilen yazılım sonradan çıkabilecek eksiklikler ve aksaklıklar için bir yazılım bakımı aşamasına ihtiyaç duyar. Bu aşamada eksik ya da hatalı görülen işler düzeltilir.

Bu süreçten sonra zamanın getirdiği etkilerle farklılaşan yeni isterler için sistem bakımı aşamasına gerek duyulabilir. Yeni gelen isterler varsa incelenip tekrardan bir yaşam döngüsü içerisinde değerlendirilir.

Proje Yaşam Çevriminin Avantajları

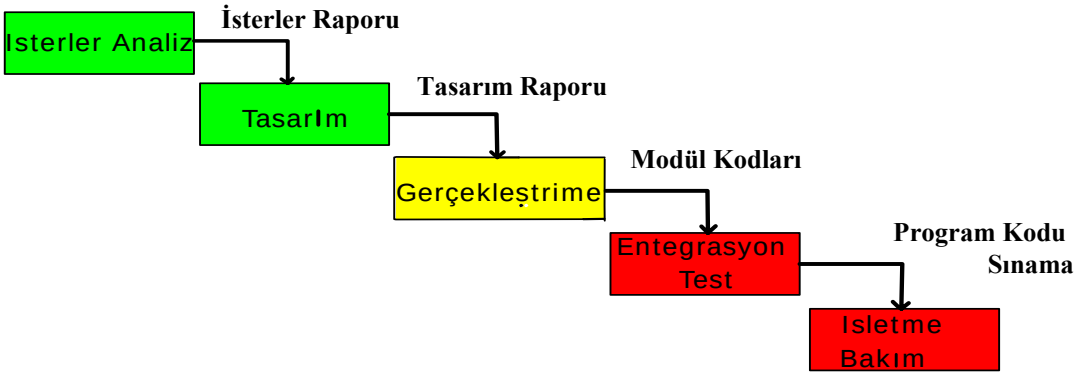
- Projeyi geliştirirken yaşam çevrimi kullanılması sayesinde proje ilerleyiş hızında büyük bir artış olacaktır. Eğer bir yöntem kullanılmıyorsa ve yazılım geliştirilmesi sadece kişisel tercihlere dönükse projenin gelişimi de çok zor olacaktır.
- Kişisel yöntemler kullanılarak geliştirilen yazılımlarda, personel değiştiğinde yeni gelen personelin sisteme uyum sağlaması süresi, proje yaşam çevrimi olmayan sistemlere oranla çok daha uzun sürecektir.
- Bütün sistemlerde olduğu gibi bilgisayar sistemleriyle oluşturulan yazılım projelerini başlangıç ve bitiş tarihleri vardır. Bu sayede tarihlerin kontrol edilmesi de kolaylaşır.

1.18 Metodolojilerden Örnekler

Aşağıda yazılım geliştirme modellerinden örnekler verilmiştir.

1.18.1 Çağlayan modeli

En eski, en tanınmış, en temel modeldir. Bu modelde oluşturulacak sistemlerin her birini bir proje olarak ele almak gerekmektedir.
Bu model bazı hükümet standartlarına girmiştir. Baştan tanımlanmış sistemler için daha çok tercih edilmektedir. Yazılımın üretilmesi aşamasında, yazılımcı ile müşteri çok sık bir araya gelmediği durumlarda uygulanan bir modeldir. Müşterinin istekleri farklılaşmayacaksa yani müşteri istekleri esnek değilse bu modelin kullanılması uygundur. Burada "Requirements Paradox"a dikkat edilmelidir. Yazılımın sağlıklı ve kaliteli olması için isterlerin sabit ve kararlı olması gerekir.[19]



Şekil 1.2 Çağlayan Modeli

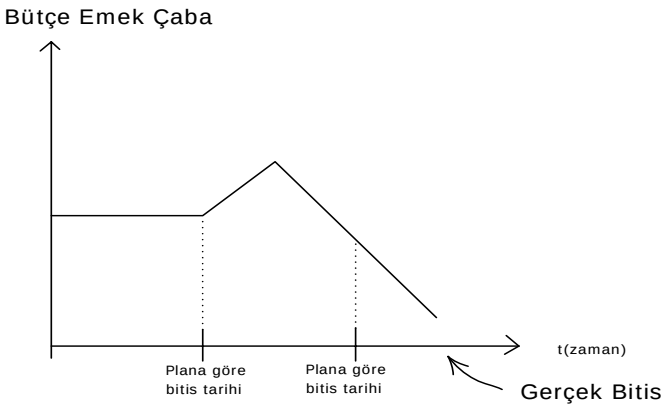
- **İsterler:** Programın yapmasını istediğimiz işleri ve diğer nitelikleri tanımlar.
- **Tasarım:** Tasarımda global tasarım ve ayrıntılı tasarım yapılır Modüllere ayrılma işlemleri yapılır Modüller tanımlanır. İlişkiler oluşturulur veri tabanı tasarımı yapılır, çıkarılan algoritmaların açıklaması yapılır.
- **Gerçekleştirme:** Bu kısımda program istenen istekler doğrultusunda ve yapılan tasarımlar ile yazılır. Hazır program parçaları kullanılıyorsa bu kodlar adapte edilir, modüller yazılır
- **Entegrasyon ve Test**: Yazılan modüller birleştirilerek aralarında olabilecek uyum sorunları ortadan kaldırılması için test yapılır.
- **İşletme ve Bakım:** Sistemi kurma, kullanma hataları giderme, ek yetenek kazandırma ve işletme

Çağlayan Modelinin Dezavantajları:

- Gerçek Yaşamda Bir İş Kesin Bitmez
- Değişiklikler Sorun Olur
- Müşteri İsterlerinin Hepsi Baştan Alınamaz(Çağlayan Modeli Bunlara Uyum Sağlamakta Zorlanır)
- Çalışan bir ürünün ortaya çıkması geç sürer, müşterilerin erken ürün görmeleri mümkün olmaz
- Sorunlar geç anlaşılır, hata düzeltme maliyeti sorunların geç anlaşılması nedeniyle yükselir.

- Yatırımcılar boş yere beklemek zorunda kalır.
- Her aşamanın bitirilmesiyle karar verme işi (doğrulama [verification]olarak adlandırılır. Toplantıda ve raporlarda kriterlerin yerine getirilmesine bakılır.)

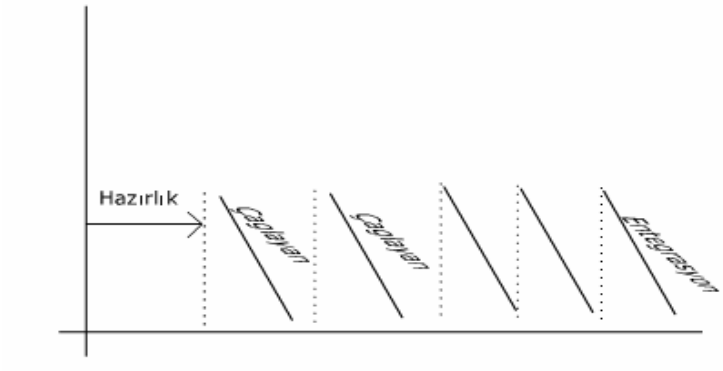
Çağlayan modelinde programın bitiş tarihi genelde ön görülenden bitiş tarihinden sonraki bir tarihte biter.



Şekil 1.3 Porje Bitiş Süresi Tahmini

1.18.2 Evrimsel metot

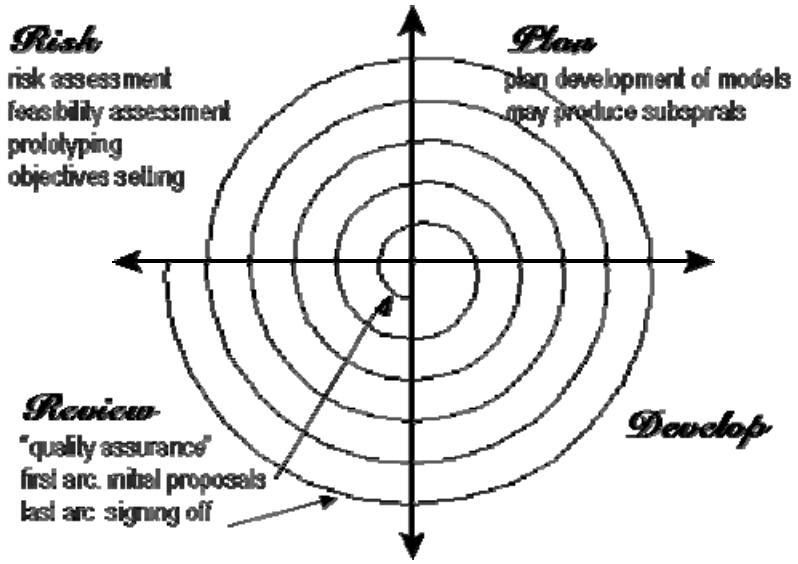
Evrimsel metot çağlayan metodunun geliştirilmiş bir şeklidir. Gerçekleştirilecek büyük bir proje küçük parçalara ayrılır. Parçalara ayrılan sistem yavaş bir şekilde gerçekleştirilir.İsterler tasarım kodlama sınama aşamaları paralel veya teker teker gerçekleştirilir. Her parça için küçük çapta bir çağlayan modeli uygulanır.[19]



Şekil 1.4 Evrimsel Model

Sistem deneme kullanma ile yavaş yavaş büyür.

1.18.3 Boehm spiral modeli



Şekil 1.5 Bohem Spiral Modeli

Çağlayan modelinin ve prototipleme yaklaşımının birleşmiş şekli olarak düşünülebilir.

Proje dört ana başlığa ayrılır ve her başlık kendi içerisinde daha detaylı bir bakış açısıyla çözülmeye çalışılır. Her bir aşama için planlama geliştirme risk çözümleme, üretim ve kullanıcı değerlendirmesi yapılır. Risk analizi, gözlem, yaşam döngüsü planı, seçeneklerin değerlendirilmesi kısımları vardır. Her aşamada bir prototip oluşturulur. Bu prototipler geliştirilerek son prototip ortaya çıkartılır. [9]

1.18.4 EVO Modeli

Tipik bir haftalık çevrimle çalışır. Her hafta yapılacak işler belirlenir. Hafta başı isterlerin analizi yapılır, görevler belirlenir. Hafta sonuna kadar verilen bu görevler gerçekleştirilmeye çalıştırılır. Hafta sonunda da sonuçların doğrulanması ve değerlendirilmesi işleri yapılır. Yeni gelen isterler doğrultusunda EVO modeli tekrar kullanılır.[19]

Her çevrim sonucunda mutlaka bir ürün teslim edilir.

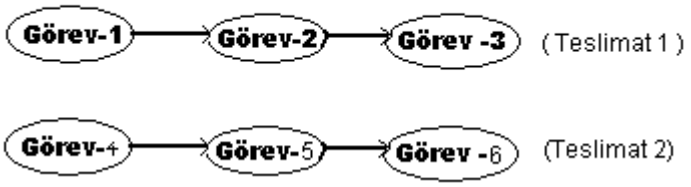
- Temel Çevrim 1–3 hafta arası sürer
- Stratejik Hedefler 1–3 ay arası
- Organizasyon hedefleri 3–12 ay arası

Zaman Kutulama (Time Boxing) ve özellik kutulama işleri yapılır.

Zaman kutulama = Belirtilen zaman diliminde hangi özelliklerin belirleneceği

Özellik kutulama = İstenilen özelliklerin gerçekleşmesi için gereken zamanın belirlenmesi.

EVO'da müşteri her çevrim sonucunda bir ürün parçası alacaktır.



Şekil 1.6 EVO modeli

Evo modelinde işler istendiği gibi gitmediği takdirde içerik kısaltılır erteleme olmaz.

2. Yazılımda Kalite ve Süreç

Yazılımların hem özel hayatımızda hem de iş hayatımızda yerini giderek arttırmasıyla birlikte daha geniş kapsamlı yazılımlar üretilmiştir. Bu da yazılım mühendisliğinin yerine getirmesi gereken işlerin doğru orantılı olarak artması anlamına gelmektedir. Yazılımlardaki artışla birlikte üretilen yazılımların daha karmaşık bir yapıda sahip olmaları, üretilen yazılımların nasıl ve ne şekilde üretileceği sorusunu akla getirmiştir. 1968 yılında ile kez ortaya çıkan yazılım mühendisliği kavramı ile birlikte o günden bu güne yazılımların nasıl ve ne şekilde yapılacağı sorusu üzerinde durulmuştur.

Yazılım üretimi işi için; yazılım geliştirme metodolojileri, programlama dilleri, çeşitli programlama araçları geliştirilmiştir. Bu gelişmelerin sağlanmasında pek çok mesleki kuruluş ve akademik kurum ortaklaşa çalışmıştır. Yazılım kalite ve süreç konusunun iyi anlaşılabilmesi için yazılım kalite ve süreç ile ilgili kavramlara bakmak gerekmektedir.

Yazılım mühendisliğinin temelleri, yazılım mühendisliğinin ürettiği ürünlerin niteliklerini anlatan teorik, bilimsel , matematiksel temellerden ve öngörülebilir sonuçlar üreten ana ilkelerden oluşur. Buradaki ana nokta, kaynakları belirlenmiş bir amaca dönüştürmek için mühendislik tasarımı ve mühendislik bilimi uygulanarak en uygun modellemenin yapılabilmesidir.[20]

Yazılım Kalitenin Tarihsel Gelişimi

Aşağıda yazılım kalitenin tarihsel gelişimi yıllara göre verilmiştir.

1924-İlk modern istatiksel kalite kontrol –Shewhart

1940-Modern istatiksel kalite kontrol kullanılmaya başlıyor

1946-İlk Bilgisayar(ENIAC)

1950-Juran,Deming ve Feigenbaum'un önderliğinde Japonya

1960-Kalite kontrol ve yönetimi felsefesi Japon ulusunun en önemli işi olarak algılanıyordu.

1968-NATO yazılım Mühendisliği Konferansı

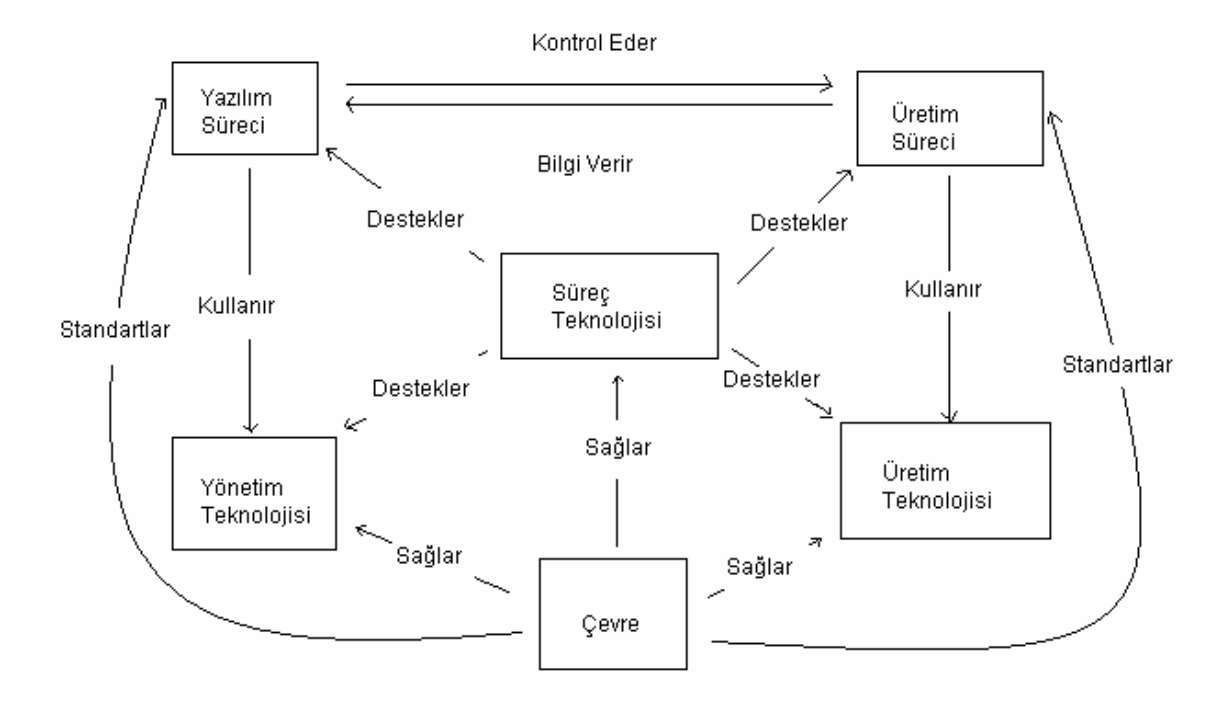
1969-İlk uluslar arası Kalite konferansı –Japonya Tokyo
1970-Yazılım Kalitesi Terimi ilk kez kullanılıyor.
1980-Batılı firmalarda toplam kalite çalışmaları yapmaya başlıyorlar.

2.1 Süreç (Process)Nedir?

Süreç kısa tanımıyla bir işi yapma yoludur. Genelde bir ya da birkaç adımdan oluşur. Bir süreci iyi tanımlamak, o sürecin amacını, ürünlerini, aktörlerini tanımlayarak olur. Eğer bir süreç yeteri kadar iyi bir şekilde tanımlanmaz ise hatalar belirlenemez. Başka bir deyişle süreç olgunlaştırılmaz ise ürün kalitesi sadece tesadüflere kalır. Süreç yaklaşımı özellikle yazılım için daha da geçerli ve önemlidir. Çünkü yazılım ancak çalışmakla anlaşılabilir ve her koşulu yazılma deneme gibi bir imkânımız olmadığı için yazılımda kalite olayının üzerinde daha ciddiyetle durmak gerekir. Sürecin amacı, bir standardı oluşturmaktır ve bu sayede değişkenliği azaltmak iyileşmeyi kolaylaştırmaktır.

Yazılım süreçlerini iyi bir şekilde belgelemek gerekmektedir. Bu belgeleme, yazılı veya grafiksel olabilir. Bir süreç başka bir yerde tekrardan oluşa bilir.Yani süreçler tekrarlı olabilir. Bir diğer unsur da yazılımların oluşturulmasında süreç olgunluğudur. Süreç için çok önemli olan "Süreç yönetimi" de bir başka önemli unsurdur. Süreç yönetimi ve kalite yönetimi bir birini sürekli olarak desteklemesi gereken unsurlardır. [9]

"Bir ürünün yazılım için kalitesi onu incelemekle belirlenmez büyük oranda onu üreten organizasyon ve süreçlerine bağlıdır."



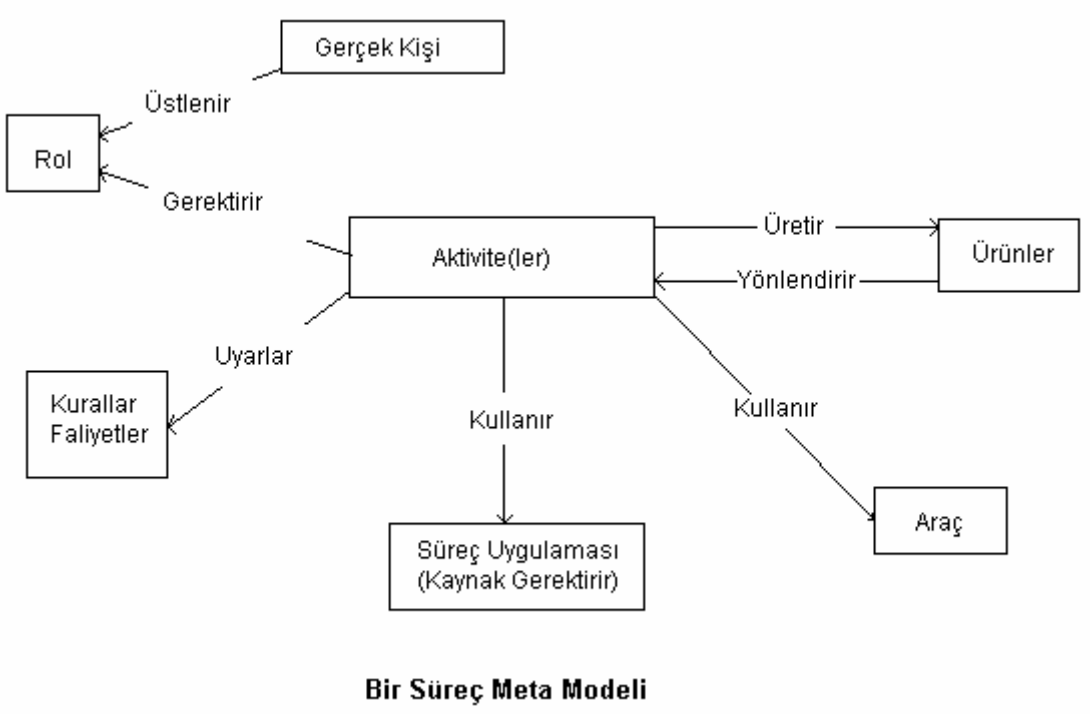
Şekil 2.1 Süreç Teknolojisi ve Süreç Yönetimi

Süreç teknolojin amacı bir süreç modelinin ortaya koymak ve tüm yazılım üretimini bu modele göre değerlendirmektedir. Süreçler sürekli gözden geçirilmeli ve iyileştirilmelidir.

2.1.1 Süreçlerin Ortak Özellikleri

- Bir işi yapma yoludur.
- Alt süreç ve adımlardan oluşur.
- Yazılı ve grafiksel belgelenmiştir.
- Tekrarlanmak üzere yapılmıştır.
- Amacı vardır.
- Girdileri ve çıktıları vardır.
- Genel ama değişkenliği azaltmaktır.
- Ölçmeyi ve iyileşmeyi sağlar.

Bir süreç meta modeli



Şekil 2.2 Süreç Meta modeli

Süreç meta modelindeki kaynak; insan, para, yer, malzeme, araç, yazılım, bilgi ve zaman olarak değerlendirilebilir.

2.1.2 Süreçlerde dikkat edilmesi gerekenler

Süreçler her şeyi kapsayacak durumda olamazlar. Süreçlerin gelişmesi ve olgunlaşması zamana bağlıdır. Mükemmel süreç yoktur. Her zaman daha iyi bir süreç olacaktır. Süreçler tutarlı, takip edilebilir, ölçülebilir ve eğitim verilebilir olmaları gerekmektedir. Bu olgunluğa varmaları için belli bir sözlük çerçevesi içerisinde anlaşılabilir terimler kullanılarak süreçlerin yazılı bir hale gelmeleri sağlanmalıdır. Süreçlerin yazılı halde uygulandığı şekilde gerçekleştirilmesine dikkat edilmelidir. Bu hususa dikkat edilmez ise süreçlerin ölçüm değerleri hiçbir anlam ifade etmeyecektir.

2.2 Süreç Modelleri

Yazılım projelerini gerçek hayata geçirirken çeşitli süreç modelleri kullanılır. Bu süreç modelleri daha önceki bölümde gördüğümüz şelale modeli, evrimsel model, spiral süreç modelleri olabileceği gibi çevrim süresini azaltmayı hedefleyen hızlı uygulama geliştirmeyi sağlayan RAD (Rapid Application Development) modeli de olabilir. Bu konuda Şelale ve Bohem Spirali modellerinin artılarını kullanan yöntem olan kilometre taşı temelli geliştirme (Milestone-Based Development) modeli veya işlerin paralel olarak yapıldığı (Concurrent Development Process Model) süreç modelleri örnek olarak verilebilir.[21]

Bir yazılım esnasında hangi yöntemin ne şekilde hangi projeler için kullanacağına yazılım firmasındaki bu işler üzerine deneyimli olan kişilerin karar vermesi gerekir.

2.3 Yazılım Kaliteye Giriş

Bilgisayar yazılımlarının istenilenleri düzgün bir biçimde yapabilmesi için tüm öğelerin uygun bir şekilde olması gerekir. Sistemlerin düzenli çalışabilmesi de ürün veya hizmetin beklenilen istekleri yerine getirip getiremediğiyle doğru orantılıdır. Bu noktada kalite anlayışı çok önemli bir yer tutar. Latince bir kelime olan "Qualis" kelimesinden türemiş ve "Qualitas" adını alan kalite sözcüğü; genel ifadeyle belli şartlar içerisinde belli bir zaman süresi içinde istenilenlerin düzgün bir şekilde yerine getirilmesidir. Yazılımlar için belirli unsurlar vardır. Bunlardan biri de kalitenin; zincirin halkalarında oluğu gibi zincirin en zayıf halkanın sağlamlılığına bağlı olduğudur. Yazılımda amaç yüksek kaliteye sahip yazılımlar üretmektir.

Yazılımda kaliteye baktığımızda ise, kullanıcının isterlerine göre açıkça belirlenmiş tanımlanan ihtiyaçları çözümüne göre kendini düzenleyebilen, kullanıcı isterlerini karşılanabilen, kullanılan çeşitli standartlara sadık yüksek güvenlikli, her işin neden ve niye yapıldığına önem gösteren, tanımlı yazılımlar üretmeyi amaçlamaktadır.

Gereksinimlerin ve isteklerin karşılanması asıl amacı ürünü oluşturmaktadır. Belirlenen standartlardan uzaklaşılması ise kalitenin düşmesine veya kalitenin olmamasına neden olur. Tabi ki belirlenen standartlar uyarınca geliştirilen yazılımlar diğerlerine oranla çok daha güvenilir ve kaliteli olacaktır. Yazılım ürünlerini gözden geçirerek konmuş olan usul ve standartlara uyulup uyulmadığının değerlendirilmesi gerekir. Kalite için belirlenen özelliklerin yeterliği sağlaması için gereken planlı etkinliklere de kalite güvencesi“quality assurance” denir.[22]

Yazılım kalitesini belirleyen temel özellikleri iki belli başlı kısımlara ayırabiliriz.

1. Yazılımın kullanımına ilgili özellikler
2. Yazılımın geliştirilmesine ilişkin isterler

2.3.1 Yazılımın kullanımına ilgili özellikler

Yazılımın kullanımındaki en önemli özellik kullanıcının isteklerini kullanıcının isteğine uygun bir şekilde sağlanmasıdır.

- **Kolaylık:**Yazılım kullanıcının tüm isterlerini karşılayabilmelidir. Ama bu isterleri karşılarken müşterinin bu işleri kolay menülerle basit bir şekilde yapmasına imkân sağlamak gerekmektedir. Menülerin estetik bir görünüme sahip olması, yazılımın kolaylığını ve karakteristiğini artıracaktır.
- **Doğruluk:**Yapılan yazılımların müşterilerin ihtiyacı olan verileri düzgün bir şekilde oluşturarak hatasız bir şekilde müşteriye verilmesidir. Bu sayede müşteriler bu yazılımı güvenle kullanabilirler.
- **Sağlamlık:**Yazılım oluşturulurken her türlü şart koşulları ve imkânlar dâhilinde ele alınmalı ve farklı şartlar altında yazılımın düzgün bir biçimde yapısının korunması sağlanmalıdır.
- **Verimli çalışma:**Yazılım çalıştırıldığında istenilen cevaplar kısa sürede elde edilmeli ve bu sonuçları elde etmek için kullanılan kaynaklar mümkün olduğunca az olmalıdır.
- **Korunmalı olma:**Yazılımı geliştiren kişilerin bilgi dâhili olmadan yazılım ya da yazılımın ihtiyacı olan veriler üzerinde değişiklik yapılmamalı, veriler elektronik ortamda korunmalı.
- **Ekonomiklik:**Yapılacak ya da yapılan yazılımlar müşteriye büyük bir külfet getirmeden sürekli olarak kullanılabilir olmalı ve imkânlar dâhilinde değişik donanımlarla da kullanılmalı [13]

2.3.2 Yazılımın geliştirilmesine ilişkin isterler

Yazılımlar kullanıma başladıklarında yazılımı kullanan kişiler tarafından yeni isteklerin oluşması doğaldır. Bu istekler yeni oluşan durumlardan veya yazılım ile yapılanlar tam olarak anlaşıldıktan sonra olabilir. Yazılımcıların bu duruma hazırlıklı olarak sistemleri tasarlamaları gerekir. Yapılan yazılımların bu özellikleri sağlaması için aşağıdaki özelliklere sahip olmalıdırlar.

- **Bakım kolaylığı:** Bakım kolaylılığının sağlanması için yazılımın oluşturulması ile ilgilenen bütün kişilerin yazılım oluşturulması aşamasında bütün süreçleri iyi belgelemiş olmaları gerekir. Ayrıca. kaynak kodlarında yeteri kadar açıklama bulunması şarttır.
- **Genişleyebilirlik:** Oluşturulan yazılımın her zaman kullanıcı isteklerine göre kendisini şekillendirmesi gerekmektedir. Tabi ki burada kullanılacak yöntemler bu işin doğru yapılabilmesi için anahtar görevi üstlenmektedir. Kullanılan yöntemler sayesinde yazılıma yeni işlevler uygun şekilde kazandırıla bilir. Ancak yazılımların büyüklükleri arttıkça bu iş giderek zorlaşmaktadır.
- **Doğrulanabilirlik:** Doğrulanabilirlik, yazılımların çalıştırılmasıyla üretilen verilerin doğru çalışıp çalışmadığını test edilebilmesini sağlar. Eğer bir yazılımın testi mümkün değilse o zaman o üretilen yazılımın doğruluğundan da emin olunamaz.

2.3.3 Yazılımların değerlendirilmesi

Yazılımların değerlendirilmesiyle ilgili olarak dikkat edilmesi gereken ana özellikler aşağıda listelenmiştir. Üretilen yazılımların bu özellikleri ve bilgileri sağladığından emin olunması gerekmektedir.

- **Denetlenebilirlik:**Yazılımın standartlarına ne derece uyduğu.
- **Doğruluk:** İsterleri eksiksiz yerine getirme kapasitesidir. Bu özellik aynı zamanda başarı için de önemlidir.
- **Hassaslık:** İşlemler sonucunda çıkan verilerin sayısal olarak doğruluk derecesi.
- **Veri Aktarımı:** Sistemin kullanacağı verilerin dış ortamdan düzgün bir biçimde alınıp alınamadığı konusunu kapsar.
- **Verilerin yaygınlığı:** Yazılım karmaşasını engellemek için verilerin standart bir yapıda olması gerekmektedir. Standart veri tiplerinin kullanılması şarttır.
- **Bütünlük:** İstenen bütün işlerin bir arada gerçekleştirilmesidir.
- **Büyüklük:** Yazılımı oluşturan kaynak kod satır sayısı, modül sayısı, öz kaynak gereksinimi gibi değerlerdir.

- **Tutarlılık:** Yazılımın üretim aşamasında tasarım ve belgeleme için aynı tekniklerin kullanılması.
- **Hata dayanıklılığı:** Oluşan bir hatanın hangi sistemlerden etkilediğinin tespiti ve hatanın büyüklüğüne karşı sistemin sağlamlığıdır.
- **Genişleyebilirlik:** Yazılım mimarisinin istekleri karşılama derecesi
- **Müşteri Memnuniyeti:** Yapılan yazılımın müşterinin ihtiyaçları ne derece karşıladığı ve bu ihtiyaçları karşılanmasından sonraki müşteri tatminidir.
- **Belgeleme:** Yazılım üretimi aşamalarında belgelemenin açık ve anlaşılır şekilde yapılıp yapılmadığı
- **Değişik Yapılarda Çalışabilmesi:** Üretilen yazılımın mümkün mertebe değişik sistemler üzerinde çalışabilmesi
- **İzlenebilirlik:** Programın içerisinde kendi hatalarını kendi mekanizması ile gösterebilmesidir.
- **Modülerlik:** Bir arada olan programın kendi iç yapısının işlevsel olarak ayrılabilir olması. Bu özellik sistem içerisinde sağlanıyorsa, belli bölümlerdeki yeni istekler için sadece o bölümle ilgili yerlerin değiştirilmesi gerekir.Bu sayede isterlerin daha kolay bir şekilde gerçekleştirilmesi sağlanabilir.
- **Kullanım Kolaylığı:** Kullanıcı ara yüzlerinin ve programın kullanımının kolay olması
- **Bakım Kolaylığı:** Sisteme sonradan ilave edilecek geliştirici ve iyileştirici kısımların kısa sürede yapılabilmesi.

3.Yazılım Metodolojileri

Bu kısımda yazılımda süreç geliştirme ve iyileştirme modellerinin neler olduğu konularından bahsedilecektir. Bu modellerin bir birlerini nasıl etkiledikleri ve yazılım geliştirme işlerine nasıl bir bakış açısı ile yaklaştıkları üzerinde durulacaktır. Yazılım süreç geliştirme modellerine bakacak olursak, ISO 9000–3, AQAP 150/160, TICKIT, TRILLIUM, CMM, ISO 12207, ISO 15504 (SPICE), BOOTSTRAP, CMMI gibi yazılım geliştirme modelleri olduğunu görürüz.

3.1 ISO 9000

Kalite sağlanma çalışmaları zaman içerisinde kalitenin oluşması, kalitenin kriterlerinin belirlenmesi ve bu kriterlerin kontrol edilmesi aşamalarından geçmiştir. Bu aşamaların tamamlanmasından sonra belli kalite standartları oluşur. ISO 9000'de bu kalite standartlarından biri olarak ilk kez 1987 yılında ISO 9000 serisi standardı olarak düzenlenmiştir. ISO 9000 serisi, her türlü endüstriyel ürün ve hizmet için kalite standart dizisidir. ISO (Internatrional Standardizasyon Organization) tarafından uluslararası standart organizasyonunun yayınladığı ISO 9000 serisi kalite sistem ve güvence standartları olarak ta tanımlanabilir. [23]

ISO'nun tarihsel gelişimine baktığımızda ise G7 diye tarif edilen ülkelerin daha kaliteli ürün üretme istekleri sonucu oluşturulmuştur. Özellikle Avrupa'da büyük ilgi gören ISO serisi, malların ve hizmetlerin kaliteli olunması, üreticilerle tüketiciler arasındaki iletişimin daha kolay bir şekilde uygulanmasını amaçlamaktadır.1987'deki birinci seri standartlardan sonra 1994 yılında ikinci seri standartlarını belirlemiştir. ISO standartları kendi içlerinde çok çeşitli olmakla beraber beş ana ISO standardından bahsedilebilir.[24]

- ISO - 9000
- ISO - 9001
- ISO - 9002
- ISO - 9003
- ISO - 9004

ISO 9000 kaliteyi yönetmek için oluşturulmuştur. ISO 9000–3 ise yazılımda kalite sistemini gerçekten yapılıp yapılmadığını incelemek için oluşturulmuş bir kalite standardıdır. Yazılımdaki kalite için ISO 9001 standardı da kullanılabilir. Fakat denetlemeyi kendi içlerinde gerçekleştiremez. Bir denetim birimine ihtiyaç vardır. Bu standartlar, firma düzeyi hakkında çok detaylı bilgiler vermekten uzaktır. Sadece genel bazı bilgiler elde edilebilir.[25]

ISO 9000 belgesi, önemli ölçüde uluslararası geçerliliği olan bir belgedir. ISO 9000 standartlarını alabilmek için firmalar ülkelerin akredite olmuş kurumlarına başvurmalıdırlar. Bu başvuru sonucunda yapılan incelemeler sonrası yerine getirmeleri gereken koşulları tamamlayan firmalar ISO 9000 belgesi alabilirler.

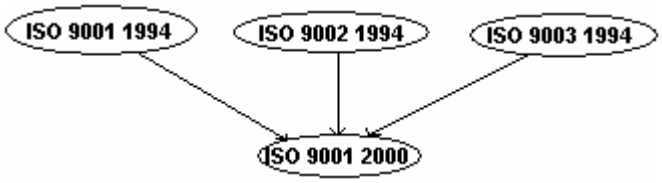
3.1.1 ISO 9000 belgelendirmesi nasıl olur?

ISO kalite standartları almak isteyen firma, bu işi bir dış denetleme kurumuna verip belge alabilir. Bu denetleme işleri firmaların kendi bulundukları yerde yapılır. Denetleme sonucunda sertifika almaya hak kazanan firmalar yapılan denetleme içeriğine göre ISO 9001, ISO 9002 veya ISO 9003 gibi sertifikalar ile ödüllendirilir. Sertifikaların sahip olunma süresi yaklaşık sertifikadan sertifikaya değişmekle birlikte ortalama 3 yıldır. Bu süre içerisinde her 6–12 ayda bir denetleyiciler tarafından firma denetlenip sertifikanın geçerliliği onaylanır.[26]

3.1.2 ISO 9001:2000

ISO 9001:2000 kendisinden daha önce çıkan ISO 9001:1994 imalat sektörü için geliştirildiğinden, yazılım üretimi ortamında yetersiz kalması üzerine geliştirilmiştir. Sahip oldukları kaliteyi kanıtlamak için firmaların başvurdukları yöntemlerden biri de kendi iç süreçleri tanımlamaktır. ISO 9001:2000 de bu amaca uygun olarak tasarım, geliştirme, üretim, test gibi aşamalar tanımlanır. ISO 9001:1994 yapısının yetmemesi dolayısıyla oluşturulan ISO 9001:2000 yazılım üretimine kolaylık getirmiştir. ISO 9001:2000 in temelinde süreç belirlenmesi, süreç yönetimi, süreçlerin takibi, süreç geliştirme aşamalarının rolü çok büyüktür. İnsan emeğinin büyük bir rol oynadığı yazılım üretme işinde iş verimini önemli bir şekilde etkisi altına alan imkanlar, motivasyon, iş şartları ISO 9001:2000 içerisinde önemle ele alınan konulardan bazılarıdır. Bu ve bunun gibi avantajlarından dolayı ISO 9001:2000 yazılım geliştirilme işlerinde yararlanılan bir standart olmuştur.[27]

ISO 9001:2000'in Düzenlenmesi



Şekil 3.1 ISO 9001:2000 Düzenlenmesi

3.1.2.1 ISO 9001:2000 Standart Bölümleri

ISO 9001:2000 standardı, ürün geliştirme, ürün tasarımı ve ürünün müşteriye verilmesi aşamalarından oluşur. Firmaların yeterliliğinin kanıtlaması gereken durumlarda alınması şarttır. Bu sayede ürün geliştiriciler ürettikleri ürünlerin kalitesini ölçebilirler. [28]

ISO 9001:2000 toplam 8 bölümden meydana gelir;

1.Kapsam

2.Referanslar

3.Tanımlar

4.Kalite Yönetim Sistemi

5.Yönetim Sorumluluğu

6.Kaynak Yönetimi

7.Ürün Gerçekleştirme

8.Ölçme, Analiz ve İyileştirme

Dördüncü maddeden itibaren karşılanması gereken şartlar başlar.Bunlara daha detaylı bir şekilde bakmamız gerekmektedir.

- **4.Kalite Yönetim Sistemi:**Süreçlerin birbirleri ile olan ilişkilerinin açıklandığı kısımdır. Süreçlerin optimize gerektiğinden ve sonuçların ölçülmesi gerektiğini ortaya koyar. Ayrıya bu bölüm dokümantasyonun ve kaynakların belirlememesinin önemini de ortaya koyar.
- **5.Yönetim Sorumluluğu:**Yönetimin kaliteyi nasıl koruyacağı bu bölümde incelenmektedir. Üst yönetim kalite yönetiminin tamamı üzerinde aktif alarak rol alıp kaliteyi tüm birimlerin nasıl koruyacaklarını alt birimlere anlatır. Bu bölümde kalite hedefleri belirlenir. Belli planlar oluşturulur. Sorumluluklar ve yetkilerin tasnifi yapılır.
- **6. Kaynak Yönetimi:** İstenen ürünlerin istenildiği zamanda oluşturulabilmesi için gerekli olan personel, donanım, malzeme gibi ihtiyaçların temin edilmesinin belirlendiği bölümüdür.
- **7.Ürün Gerçekleştirme:**Ürün gerçekleştirilmesi için zorunlu olan süreçler tasarım ve geliştirme faaliyetleri bu bölümde planlanmalı ve her bir sürecin kontrolleri yapılmalıdır. Müşterilerden ürün ile ilgili detaylı bilgi alınarak müşterinin talepleri göz önünde bulundurulmalıdır.

• **8. Ölçme, Analiz ve İyileştirme:** Kalite yönetimi için ölçme ve analiz kullanılarak standartlar konulmalıdır. Bu ölçümler neticesinde devamlı olarak daha iyiye doğru bir yönelim gelişecektir.

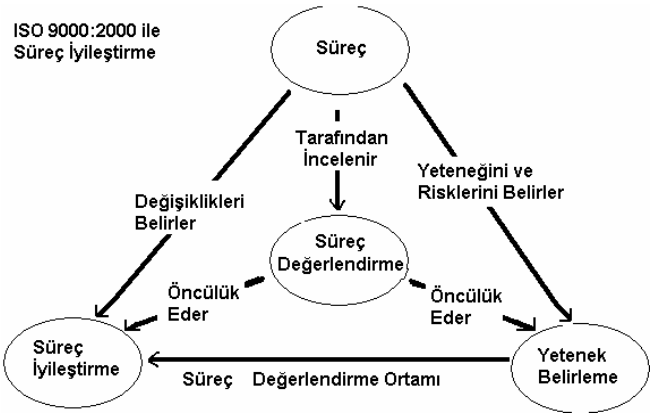
Müşterilerin ürün hakkındaki endişeleri veya memnuniyetleri ölçülerek müşterilere daha iyi bir ürün sunma yoluna gidilmeli. Ürünü oluşturan adımlar gözden geçirilerek daha etkin bir şekilde bu süreçleri nasıl geliştiririz sorusu üzerinde durulur. Oluşan hataların kaynağına gidilerek daha sonra bu hatanın oluşmaması için gerekli düzenlemelerin yapılması gerekliliği üzerinde durulmalıdır.

3.1.2.2 ISO 9001:2000 Süreçleri Geliştirme

Bir şirketin mevcut şartlarda ürettiği malların kalitesini yükseltmek için;

• Süreçler incelenerek şu andaki niteliklerini değerlendirecek bir metot ya da metodolojilere ihtiyaç vardır.

• Kullanılacak bu metotlarla elde edilecek sonuçların, süreçleri iyileştirme çalışmasının bir bütünü olması gerekmektedir.



Şekil 3.2 ISO 9000:2000 Süreç iyileştirme

3.2 AQAP

Bir NATO standardı olan AQAP (Allied Nato Quality Assurance Program), kalite güvence sistemi NATO ya bağlı askeri kuruluşların kalite standartları için geliştirilmiştir. Çeşitli AQAP standartları vardır.

AQAP-2110 (Tasarım, Geliştirme ve üretim)
AQAP-2120 (Üretim için)
AQAP-2130 (Muayene ve test)
AQAP-2131 (Son muayene için)
AQAP-2009 (AQAP 2000 Serisinin Kullanımı İçin)
AQAP-2000 (Ömür Devri Boyunca Kaliteye Bütünleşik Sistemler Yaklaşımı için)
AQAP-160 (Yazılım İçin)

AQAP standartları NATO'un kalite ve güvence için geliştirilmiş bir standarttır. Biz burada yazılımla ilgili olan AQAP 150 ve 160 standartlarını inceleyeceğiz.[30]

3.2.1 AQAP 150/ 160

AQAP 150–160 standartları, yazılımla ilgili olarak askeriyenin ihtiyaçları uyarınca geliştirilmiştir. AQAP kullanarak geliştirilen ürünler ileri teknolojinin kullanıldığı bilgi teknolojileri hizmetleri; yazılım tasarım üretimi, sistem tasarımı, sistem geliştirme, entegrasyon, savunma, iletişim yazılımları, sistem geliştirme, sistem bakım, onarım ve destek hizmet sunumu, simülatör geliştirilmesi gibi ürünlerdir. Bu ürünlere baktığımız zaman hepsinin son derece komplike yapılar olduğunu görüyoruz. AQAP kullanılarak gerçekleştirilen yazılımlar, isterler doğrultusunda tasarlanmalı ve buna göre geliştirilmelidir. NATO'ya bağlı ülkelerin kullandığı AQAP, kalite güvence işini aynen NATO'nun diğer işlerinde olduğu gibi kalite açıklık ve netlik ilkeleriyle bir birine bağlamaktadır. Ürün geliştirilmesinde başarıyı sağlamak için kalite yönetimi kurulmalı ve sürdürülmelidir. [31]

3.2.2 AQAP 160'ın genel yapısı

AQAP'ı anlamak için AQAP'ın genel yapısını incelemek gerekmektedir. AQAP'ın kendi yapısı içerisinde QAR denilen kalite güvencesi temsilcisi vardır. Bu temsilci, alıcının şartname içerisindeki yetkisi veya yetkili temsilcisidir. AQAP 160'ın genel çerçeve yapısı ise üç kısımdan oluşur. İç kısımda süreçlerle ilgili işler yapılır.İkinci kısımda süreçleri kapsayan bir yapıda kalite sistem gereksinimleri belirlenir. Dış kısımda ise NATO'un kendine özgü yapısı gereği belirlenen özel tanımlar yapılır.

Şekil 3.3 AQAP genel yapısı

AQAP'ın Türkiye'deki durumuna bakarsak dört firmanın AQAP ın yazılım geliştirme kısmı olan AQAP150–160 standardında olduğunu görüyoruz. Bunlar aşağıda belirtilmektedir.

Tablo 3.1 AQAP in Türkiye'deki Konumu

Firma Adı	**AQAP**
AYDIN YAZILIM VE ELEKTRONİK SANAYİ A.Ş.-Ankara	AQAP 150
ASELSAN A.Ş. Mikrodalga ve Sistem Teknolojileri Grubu-Ankara	AQAP 150
ALTAY KOLLEKTİF ŞİRKETİ M.MURAD DURAL VE ORTAĞI-Ankara	AQAP 150
HAVELSAN HAVA ELEKTRONİK SAN.A.Ş.-Ankara	AQAP 160

3.2.3 AQAP belgesi isteyen kuruluşların yapacakları işlemler

AQAP belgesi almak isteyen kuruluşlar, kendi ürettikleri mal ve hizmetleri, kalite yönetimine yönelik tasarımlarını, ürün geliştirme süreçlerini, üretimlerini, entegrasyon aşamalarını takip etmelidirler. Burada kullanılacak yöntemleri belirlemeleri gerekmektedir. AQAP tarafından istenecek kalite için gerekli olan rehberler AQAP'ın istediği formatta hazırlanır. Belgelendirmeyle ilgili yapılacak ön inceleme sonucunda yanlışlıklar ve eksiklikler düzeltilir. Sonradan tekrar incelenecek yanlışlık veya eksiklik var mı diye bir daha gözden geçirilir. Belgeleme sonucunda yeterli görünen kuruluşlar, kendi kuruluşlarında inceleme heyeti için firmayı hazırlarlar. Firmanın tanıtımı gerçekleştirilir. Firma AQAP belgesine göre hazırlanan evrakları ve organizasyon ile ilgili bilgileri verir. Eğer firma yeterli görülürse AQAP belgesi almaya hak kazanır.

3.2.4 AQAP belgesine bulunan kuruluşların yapmaları gerekenler:

AQAP belgesi bulunan firma belgeyi aldıktan sonra belirtilen disiplin yöntemlerini istisnasız uygulamak ve devamını sağlamak zorundadır. AQAP belgesine sahip kuruluş belgeye sahip olduğu süre içerisinde AQAP kalite sistem yapısında yaptığı değişiklik oluşmuş ise bunu belirtilen süre içerisinde kendi yapısına entegre etme sözü altına girmiştir. Yapılacak ara denetlemelerde başarılı olması gerekmektedir. Ara denetlemelerde bulunan uygunsuzluklar olmuş ise bu sorunları AQAP'ın çalışma biçiminin belirlediği şekilde çözümlemelidir. Sorunlar giderildiğinde denetleyicileri bilgilendirme yapılmalıdır. Eğer sahip olunan AQAP belgesi geçerlilik süresi uzatılmak isteniyor ise süre bitmeden üç ay önce tekrardan başvuru yapılmalıdır. Bu başvuru yapıldıktan sonra AQAP tarafından belirlenmiş bir süreç devreye girer. Belgeye sahip olan kuruluş üretimi ile ilgili sorunlar yaşıyorsa veya personel geniş çapta değiştiriyorsa ya da üretimini farklı sahalara taşıyorsa bu kuruluş bu işleri yapmadan on gün önce AQAP belgesini aldığı kuruluşa bu değişiklikleri belirtmek zorundadır. [33]

3.3 SWEBOK 'a göre yazılım mühendisliği

SWEBOK (Software Engineering Body of Knowledge)yazılım mühendisliği sınıflama araçlarından biridir. SWEBOK'a göre yazılım mühendisliği on temel başlık altında toplanır. Bunların her birine Knowledge Areas (KA) (Bilgi alanları denir.).Aşağıda bu alanlar isimleri ile birlikte verilmiştir.[34]

Tablo 3.2 Swebokun Anahtar süreçleri

Yazılım Mühendisliği Alanları SWEBOK'a göre	The SWEBOK Knowledge Areas (KAs)
1-Yazılım İsterleri	Software requirements
2-Yazılım Tasarımı	Software design
3- Yazılım İnşası	Software construction
4- Yazılım Sınama	Software testing
5- Yazılım Bakımı	Software maintenance
6- Yazılım Konfigürasyon Yönetimi	Software configuration management
7- Yazılım Mühendisliği Yönetimi	Software engineering management
8- Yazılım Süreci	Software engineering process
9-Yazılım Araç ve Metotları	Software engineering tools and methods
10- Yazılım Kalitesi	Software quality

SWEBOK ta yazılım için kullanılan dilin herhangi bir önemi yoktur önemli olan KA(Knowledge Areas) düzgün olarak kullanılıp kullanılmadığıdır C++ ya da fortran ile yazılımın yapılması önemli değildir.

Tablo 3.3 Anahtar süreç Alanları ve Dsiplinler

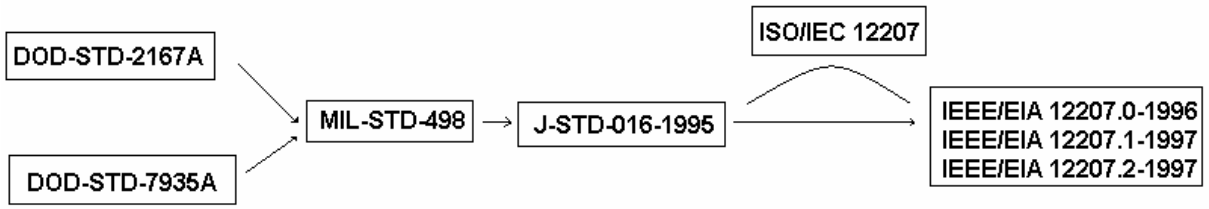
KA(Knowledge Areas) Alanları ileilgili olan disiplinler	
• Computer engineering • Computer science • Management • Mathematics	• Project management • Quality management • Software ergonomics • Systems engineering

3.4 ISO/IEC 12207

1990'ların başında ve ortalarında DOD-STD-2167A, DOD-STD-7935A,MIL-STD-498 gibi standartlar Department of Defense tarafından geliştirilmiştir. Hazırlanan bu standartlara rağmen teknolojik gelişmeler yazılımın artan dinamik yapısı nedeniyle istenilenlere karşılık verememiştir. Department of Defense in hazırladığı DOD-STD-2167A standardı, yazılım işlerinde istenilenleri yereri kadar karşılayamaması, yapısının çağlayan sistemine sıkı bir bağı varmış gibi görünmesi, resmi belgelendirmeye önem veren bir yapısının olması ve nesneye yönelik projelerde kullanım zorluğu gibi eksiklikleri vardır. Bu sorunlardan bazıları MIL-STD-498 tarafından ortadan kaldırılmıştı. Örneğin yazılım geliştirme işi içersinde birden fazla iç yapı kurarak(built) şelalenin eksikliğini belli bir oranda kaldırmıştır. Aynı zamanda mevcut yazılımlar üzerinde tekrar kullanabilme, mevcut yapıyı daha kolay bir biçimde ele alarak yapıyı daha kolay bir şekilde geliştirebiliyordu. Ama yazılım işinin ana unsurlarından sadece yazılım geliştirme kısmıyla yeteri kadar ilgilenip diğer kısımlarla yeteri kadar ilgilenmiyordu.Tam bu noktada 1987 yılında ISO, IEC(International Electrotechnical Commission), JTC(Joint Technical Commite) bir araya gelerek 1 Ağustos 1995 yılında ISO/IEC 12207 standardını yayınladı.[35]

3.4.1 ISO/IEC 12207 standardının özellikleri:

ISO/IEC 12207 standardının yazılımın satın alınmasından ürün teslimine kadar bütün süreçleri kapsayan bir yapısı vardı. Her süreçte bir agent vardır. Agent süreç sorumlusudur. Süreçlerin birbirleriyle olan ilişkileri az, süreçlerin içlerindeki ilişkiler ise fazladır. Süreçler düzeyinde tanımlamalar yapılmıştır. Bu tanımlamalarla birlikte proje içerisinde görev yapan kişiler yapılan işlerin maksadı ve yararının ne olduğu konusunda bilgilendirilir. ISO/IEC 12207 standardı yazılımı tek olarak görmez. Yazılım oluşturulması için çevre şartlarının da ne kadar önemli olduğu üzerinde durulur. ISO 12207 yapısı gereği hem askeri hem de sivil projelerde kullanılabilir. ISO 12207'in zayıf tarafına baktığımızda ise süreçleri tanımlamasına rağmen bu süreçleri ne kadar yapıldığının yeteneksel olarak değerlendirmemesini verebiliriz.



Şekil 3.4 ISO/IEC 12207 VE IEEE/EIA in standartlarının gelişim tarihi

3.4.1.1 ISO 12207 yapısı

Üç Parçalıdır:

1-Normative: Software Lifecycle Process (Yazılım yaşam döngü süreçleri)

2-Informative: Guide for 12007

3-Informative: Gerçekleştirme düşünce ve rehberi

12207 yazılımla ilgili tüm taraflara hitap eder.

Bu taraflar;

1-Teknik Personel: Yazılım Müh, Programcı, Testçi, İşletmeci

2-Yönetici Personel: Teknik Müdür, Proje Yöneticisi, Kalite Yöneticisi

3-Kullanıcı

4-Yazılım Alıcı, satıcı, sahibi

Üç Ana Guruba Hizmet Eder:

1- Yazılım geliştiren bakım yapan
2- Yazılım alım satımı, sözleşme işleri
3- Yazılım süreç yönetimi

1.Kategori: Ana Süreçler

- Edinme Süreci
- Sağlama-Tedarik Süreci
- Geliştirme Süreci
- İşletme Süreci
- Bakım

2. Kategori: Destek Süreçleri

- Belgeleme Süreci
- Konfigürasyon Süreci
- Kalite Güvence Süreci
- Doğrulama Süreci
- Geçerlilik Süreci
- Ortak Gözden Geçirme Süreci
- Denetleme Süreci
- Sorun Çözme Süreci

3. Kategori: Kurumsal Süreçler

- Yönetim Süreci
- Altyapı Süreci
- İyileştirme Süreci
- Eğitim Süreci

3.4.1 12207'nin amaçları

1- Yazılımın yaşam döngüsünün tüm süreçleri için ortak bir çerçeve oluşturmaktadır.
2- Tüm ilgili taraflar için bir çerçeve oluşturmaktır.
3- "Satın Alma " dan , kullanımdan çıkarılanlara kadar tüm süreçleri tanımlamak.
4- Dünyadaki yazılım ticaretini daha kolay bir şekilde yapılmasını sağlamak.
5- Kuruluşlara süreçlerini oluşturulmasında yardımcı olmak.

3.4.2 12207 süreçleri

1- Her süreç kendi içinde " Cohesive " kavramlıdır. Ama yine de süreçler arasında " Coupling"
vardır. En çok coupling yönetim sürecindedir.
2- Her sürecin bir sorumlusu, sahip ajanı vardır. Ayrıca party denilen bir taraf daha vardır. Party tarafı süreçte yer alan, kurum içi görevli tarafıdır. Organizasyon tarafı, süreçte yer almayan dış tarafıdır.
3-Uyarlama: Projeler arasında büyük farklar olabilir. Her projede uygulanabilen veya uygulanmayan süreçler ve aktiviteler vardır. Bu nedenler her projede nelerin uygulanacağı ve nelerin işin dışında bırakılacağı belirlenir. Bu işe uyarlama denir. Her projenin standart süreçleri, uyarlanmış süreçleridir.

3.4.3 Ana süreçler

ISO 12207'in bir süreci olan edinme sürecini örnek olarak alıp inceleyecek olursak;

Edinme Süreci, yazılım ürün veya hizmeti edinme işlerini tanımlar. Süreç ajanına edinici denir. Edinme Süreci beş aktiviteden oluşur:

- **Başlama:** Sistem ve yazılım isterleri belirlenir, tanımlanır, edinme planı yapılır.
- **Teklif isteme**:RFP (Request for Proposal) hazırlanır isterler şartname yazılır ve duyurulur.
- **Sözleşme Hazırlık**: Sağlayıcı seçimi, ihale, sözleşmeyi yapma ve imza
- **Kabul ve Sonuçlandırma :** Kabul işlemleri ,planı yazılır,uygulanır.

3.4.4 Başlama aktivitesinin görevleri

- Edici bir yaklaşım içerisinde istenilen ürün ve hizmet belirlenir.
- Edinici, sistem isterlerini tanımlar, analiz eder.
- Edinici, analiz işini başkasına yaptırabilir. Eğer böyle bir durum söz konusu ise bu seferde onaylama işini üstlenir
- Analiz işleri, geliştirme sürecine uygun yapılır.
- Edinici bazı seçenekleri değerlendirmek zorundadır:
- Hazır raf ürünü alma
- Kuruluş içerisinde yazılım geliştirme
- Sözleşme ile dış kuruluşa ihale
- Mevcut bir ürün geliştirme ya da zenginleştirme

Eğer hazır raf ürünü alınacaksa şu özelliklere dikkat edilmelidir:

- İsterle yerine getiriliyor
- Belgeleri yeterli
- Kullanım şartı, mülkiyet, lisans, garanti uygun fiyat
- Gelecekteki bakımlar planlanmış olmalı

Edici bir edinme planı hazırlamalıdır.

- Sistem isterleri
- Sistemin planlanan kullanımı
- İhale sözleşme türü
- İlgili organizasyonların sorumlulukları
- Destek yaklaşımı
- Riskler ve risk önlemleri
- Edinici kabul şartları (kriterleri) ve usulleri belirleyecektir.

3.5 IEEE/EIA 12207

ISO/IEC 12207, Amerikalılar tarafından gözden geçirilip bazı ilaveler yapılmasından sonra oluşturulmuş bir standarttır. Bu standart, ISO/IEC 12207 standardını tamamıyla kapsamakta ve ilave olarak yeni kavramlar getirmektedir. Yapısı içerisinde proje bütçesi, yazılım maliyeti, müşteri, edinici, geliştirici, alt yüklenici, yazılım geliştirme aşamalarına detaylı olarak ele alır. Kendisinden önce geliştirilmiş projelerin IEEE/EIA 12207'e geçilmesi gerektiği durumunda yazılımlara kolaylık sağlayarak uyarlanabilmesini sağlar. IEEE/EIA 12207, üç cilt halinde yayınlanmıştır. Adları ve içerikleri aşağıdaki tabloda verilmiştir:[36]

IEEE/EIA 12207.0–1996 ABD için açıklamalarıyla ISO/IEC 12207 ve ekler.
IEEE/EIA 12207.1–1997 Belgelendirme için yardımcı dokümanlar ve kılavuz.
IEEE/EIA 12207.2–1997 ISO/IEC 12207'de farklı bakış açılarına sahip alternatif uygulama yaklaşımı.

IEEE/EIA 12207 standarttın yapılış amacı, büyük ve karmaşık projeler olmasına rağmen, orta ve küçük ölçekli firmalarda uygulanabilmektedir. Belgeleme şart koşulmakla birlikte, belgelerin içeriği veya biçimi tanımlanamamaktadır. Bu standarttın etkili bir şekilde olabilmesi için üst yönetimle birlikte şirketin bütün çalışanlarının o işe gönülden bağlı olması gerekmektedir. Personel eğitimi ve şirket politikasının iyi anlatılmış olması gerekmektedir.

3.5.1 Süreçleri:

IEEE/EIA 12207 süreçleri yazılımın ihtiyaç olduğu andan sonra başlar. Yazılımın kullanılmayacağı ana kadar devam eder. Projeleri, baştan son noktaya kadar değerlendirir. IEEE/EIA 12207 yazılımın yaşam süreçlerinin tamamını bünyesi içerisinde bulundurmaktadır. Bunları üç ana süreç içerisinde değerlendirir.

Temel Süreçler: Yazılımın gerçekleşmesi için olmazsa olmaz süreçlerdir.

Destekleyici Süreçler: Süreçlerin kendilerine has işlevlerini yerine getirmesine yardımcı olan süreçlerdir.

Örgütsel Süreçler: Yaşam çevriminin oluşturulmasını, denetlenmesini ve iyileştirilmesini sağlar.

Tablo 3.4 IEEE/EIA 12207 süreçleri

Temel Süreçler	Destekleyici Süreçler	Örgütsel Süreçler
Edinme	Belgelendirme	Yönetim
Sağlama	Düzenleşim Yönetimi	Altyapı
Geliştirme	Kalite Güvence	İyileştirme
İşletme	Doğrulama	Eğitim
Bakım	Geçerleme	
	Birleşik Gözden Geçirme	
	Denetim	
	Problem Çözme	

3.5.1.1 Temel süreçler:

IEEE/EIA 12207'in yazılımın gerçekleşmesi için olmasa olmaz süreçleri olan temel süreçler aşağıda verilmiştir.

- **Edinme Süreci:**Bu süreçte, müşterinin görev ve etkinlikleri tanımlanır. Alıcının isteği sonucunda firmalardan fiyat teklifleri alınır. Yazılımı gerçekleştirecek firma tespitinden sonra yazılımı üretecek kuruluş, müşteriden isterleri alır ve kendi tanımlamalarına göre değerlendirir. Teklif isteği, sözleşme hazırlığı, sözleşme sonrası ürün kabulü, sözleşme tamamlanması gibi kısımları içerir.
- **Sağlama Süreci:**Yazılım şirketinin etkinliklerini ve görevlerini kapsar. Yazılım sözleşmesinin imzalanmasıyla başlar. Bu sürecin işlerini yerine getirmesi içinin nasıl yapılacağının belirlenmesi, kaynakların tanımlanması, planın oluşturulması, denetleme, değerlendirme, teslim ve kabul kısımlarının yapılması sürecidir.
- **Geliştirme Süreci:**Yeni yazılım geliştirilmesini ve ya var olan yazılımın etkinliğinin arttırılması kısımlarını içerir. Gerçekleştirilecek yazılım, sistemin tamamını kapsayabilir ya da belli bir kısmı ile ilgili olabilir. Geliştirme Süreci içerisinde, süreç gerçekleştirim, sistem tasarımı, yazılım mimari tasarımı, yazılım

kodlama ve test, yazılım tümleştirme, yazılım geçerlilik testi, yazılım yükleme kısımlarından oluşur.

- **İşletme Süreci:**Sistem gerçekleştikten sonra sistemin ideme ettirecek kullanıcının etkinlik sağası görevleri ile yazılımın işletilmesi ve kullanıcılara verilecek yardımları kapsayan kısımdır.
- **Bakım Süreci:**Bakım süreci sistem tesliminden sonra başlar. Bu süreç içerisinde kullanıcıya verilen yazılımın eksikleri veya hataları varsa bu eksiklikler ve hatalar ortadan kaldırılmaya çalışılır. Bakım süreci; problem tespiti, probleme karşı çözüm üretme, problem çözüm, yapılan değişikliğin sisteme entegresi aşamalarından meydana gelir.

3.5.1.2 Destekleyici süreçler: Projenin kalitesini ve içeriğini arttıran süreçler olarak tanımlanabilir. Bu süreçlerin detaylı açıklanması aşağıda verilmiştir

- **Belgelendirme süreci:**Yazılım gerçekleştirme aşamalarında üretilen bilgilerin sonradan kullanılabilmesi için düzgün bir formatta kayıt altına alınması gerekmektedir. Bu, personelin ilgi duyduğu bilgilerin planlandığı, biçimlerinin belirlendiği süreçtir.
- **Konfigürasyon yönetimi süreci:** Yazılım adımlarının belirlenmesini sağlayan bir süreçtir. Sürümlerin, denetim altında tutulmasını sağlayarak değişiklik isterlerinin raporlanmasını sağlar. Düzenleşim yönetimi, sürüm yönetimi ve teslim işleriyle de ilgilenir.
- **Doğrulama süreci:**Süreçlerin doğru bir şekilde yürütülmesi işleri ile ilgilenir. Doğrulama, sistemin yapılması için gerekli olan isterlerin tam olarak doğru olup olmadığından sorumludur. Yapılan işler sonucunda ortaya çıkan verilerle iş yapılmadan önce belirlenen verilerin bir birlerini tutup tutmadığına bakılır.
- **Geçerleme süreci:**Burada sistemin son durumu ile daha önceden belirlenmiş durumunun birbirlerini tutup tutmadığına bakılır. Projenin büyüklüğü ile geçerlemenin birbiri ile doğru orantılı bir ilişkisi vardır. Oluşturulacak sistem, ne kadar büyük ve detaylı ise geçerleme süreci de o orantıda detaylı ve geniş bir biçimde ele alınmalıdır.

- **Birleşik gözden geçirme süreci:**Alıcı ve satıcının bir araya gelerek oluşturdukları bu süreçte, ürünü geliştiren kurum ya da kuruluş gerçekleştirdikleri proje hakkında bilgi sunar. Yapılanları ve yapılacakları anlatır.
- **Denetim süreci:**Satıcının verdiği sözler doğrultusunda denetlemelerin yapıldığı süreçtir.
- **Problem çözme süreci:**Karşılaşılan sorunlarının giderilmesi için, problemi meydana getiren nedenler tanımlanır ve bu problemi oluşturma eğiliminde olan noktalar bulunarak bunların giderilmesi işleri üzerinde durulur.

3.5.1.2 Örgütsel süreçler:

Yapının işlevselliği ve bütünlüğünün sağlanması amacı ile oluşturulan örgütsel süreçler IEEE/EAI' da toplam dört adettir.

- **Yönetim Süreci:**Yazılım yaşam çevriminin en genel anlamda etkinlik ve görevlerini tanımlar. Etkinlikleri planlanmayı, denetlemeyi kapsar. Temel süreçlerle bazı yönlerden benzemesine rağmen amaç hedef ve yöntemler bakımından bazı farklılıklar görülebilir.
- **Altyapı Süreci:**Yazılımın oluşturulması süreçlerinin tamamı ve kullanılacak her türlü altyapı etkinliklerini kapsar. Altyapı içerisinde yazılım, üretim, araçlar ve yardımcı olan yan veya alt sistemler bulunur.
- **İyileştirme Süreci:**Süreçleri değerlendirip ölçerek süreçlerin denetlemelerinin yapıldığı süreçtir. Bu süreçte ana amaç süreçleri geliştirip ihtiyaç doğrultusunda bir yapı oluşturulmasını sağlamaktır.
- **Eğitim Süreci:**Bu süreç içerisinde personele gerekli teknik ve yönetimsel bilgiler verilerek personel yetiştirmek amaçlanır.

3.6 CMM (Capability Maturity Model)Yetenek Olgunluk Modeli

"CMM etkin bir yazılımın süreçlerini anahtar elemanlarını tanımlayan bir *çerçeve modeldir*; *olgun olmayan* bir süreçten *olgun* ve disiplinli bir sürece giden evrimsel bir yol çizer. CMM yazılım sürecinin olgunluğu üzerine hüküm vermek ve endüstrideki pratiklerle karşılaştırmak için bir ölçüm aracıdır." (Prof Dr Fuat İNCE Yazılımda Süreç ve Olgunluk Modelleri).

CMM'in basamaklı bir yapısı vardır. Yazılım süreç iyileştirmesi, organizasyonun stratejik planları ve iş hedefleri, yapısı, kullandığı teknoloji ve sosyal kültürü ile bağlantılı olduğunu bilen CMM buna göre yazılımı değerlendirir.

CMM'e göre:

- Yazılım, tanımlı süreçlerde yürütülmüyorsa, başarı şansa ve koşullara kalmıştır.
- Sürekli başarı, tanımlı süreçlerle olur.
- CMM, olgun olmayan bir kuruluşun, olgun bir kuruluşa giden adımları belirler.
- Yazılım süreç performansı, elde edilen sonuçlarla ortaya çıkar.
- Yetenek, olgunluğa bağlıdır, olgunlukla gelir.

Şimdi olgun olan ve olmayan kurumların özelliklerine bakacak olursak; bu konuda şu tespitler yapılabilir.

Olgun Olmayan Kurum Özellikleri:

- Yazılım süreçleri proje sırasında doğaçlanır önceden ön görülen kurallar yerine o anda oluşan duruma göre işler belirlenir.
- Süreçler var olsa bile kuvvetle işlenmez, mecbur tutulmaz anlatılır.
- Yöneticiler sürekli kriz çözmekle (fire firhting) meşguldürler.
- Kalite yükseltici etkinlikler ara sıra uygulanır.

Olgun Kurumun Özellikleri:

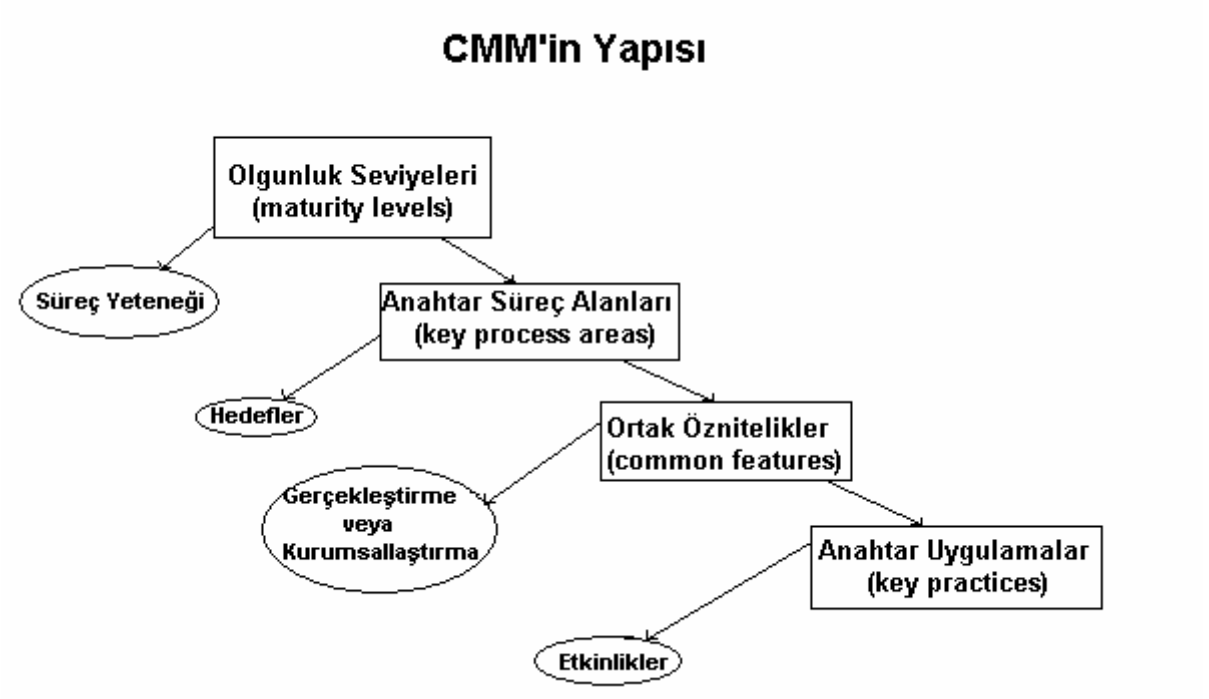
- Her iş için eğitilmiş yetenek vardır.
- Süreçler belgelenmiştir ve uygulanıyordur.
- Her çalışan, sürecin değerini anlamış, özümsemiştir.

3.6.1 CMM (Süreç Olgunluk Çerçevesi)

Organizasyon çapında yazılım sürecinin tanımlı olmadığı durumlarda, projelerde başarının tekrarı kişilere bağımlıdır. Bu durum, uzun süreli verimliliği ve kaliteyi iyileştirmeye olanak vermez.
Sürekli iyileşme ancak, etkin yazılım mühendisliği ve yönetim pratiklerinden oluşan bir süreç altyapısı inşa edilerek mümkündür. Aynı zamanda CMM olgun bir yazılım organizasyona giden *aşamaları,* öncelik sırasıyla veren bir modeldir.

3.6.2 CMM yapısı:

CMM'ın yapısı beş düzeyden oluşur. Her düzeyin kendi içerisinde tanımlanmış anahtar süreçleri mevcuttur. Bu anahtar süreçlere key proccess areas denir. Süreçlerin ortak özellikleri de vardır. Bu ortak özelliklere "common features "denir.



Şekil 3.5 CMM Yapısı

SPICE'dan farklı olarak basamaklı bir yapısı vardır. Birinci düzeyden beşinci düzeye doğru olgunlaşma yolu izler. Bu izlediği yol basamaklı bir yoldur. Her bir sonraki basamağa geçebilmesi için KPA'lar daki isterleri yerine getirmesi

gerekmektedir. CMM deki değerlendirme ise 2.düzey KPA ların dan başlar. Ortak özellikler ve KP ler değerlendirilir. Hep KP için %80 üstü geçer, altı kalır 62 KP'nin 50 sinden başarılı olması gerekmektedir.3.düzeye çıkmak için ise 50 KP den 40 tanesinden başarılı olması gerekmektedir. 4.Düzeye geçmek için 12 KP'den 10 başarılı olmalı ve son olarak 5.düzeye çıkmak için 26 KP den 21 inden başarılı olmalıdır.

3.6.2.1CMM amaçları:

Ortak özellikler:

1-Yerine getirme Taahüdü: (Committment to Perform)

Şirketin kurumsal politikalarının ve liderliğin sürece sahip çıkması üst yönetiminden yazılı tahadütüdür.

2-Yerine getirme Yeteneği: (Committment to Perform)

Sürecin başarıyla uygulanması için gerekli kaynakların, eğitimin altyapının var olması

3-Aktiviteler(Key Process): Her sürecin kendine ayit anahtar süreçleri vardır.

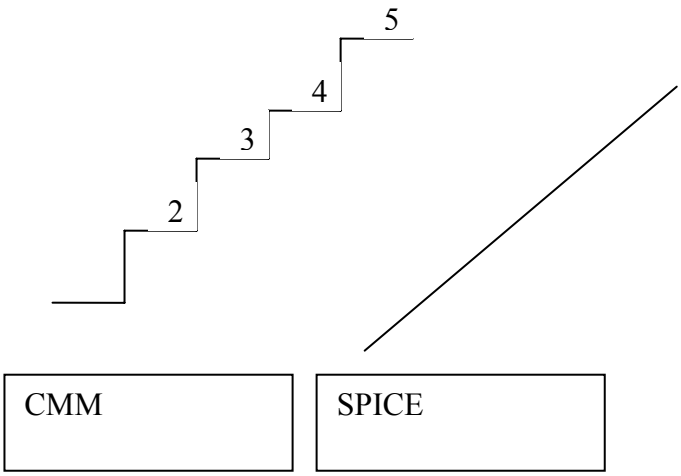
CMM 18 KPA ve 150 KP vardır.

Tablo 3.5 CMM Anahtar süreç alanları

2.Düzey	62
3.Düzey	50
4.Düzey	12
5.Düzey	26
Toplam:	**150**

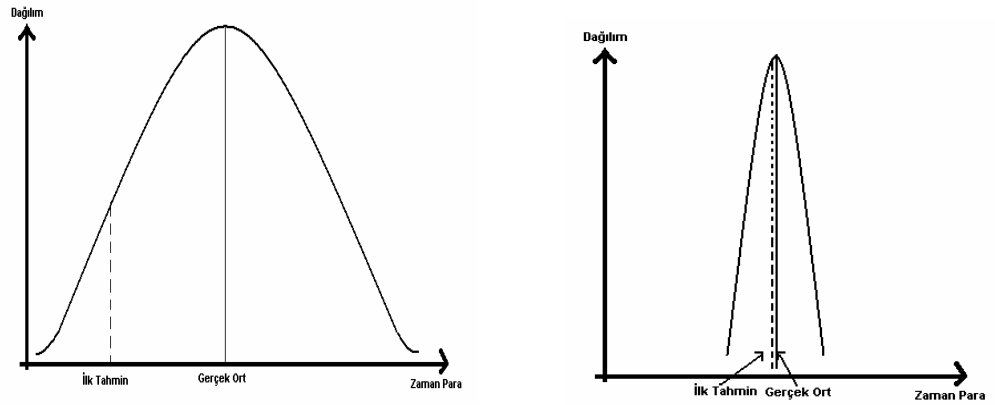
4-Ölçüm ve Analiz: Süreçlerin başarımını belirleyecek ölçüler toplanır ve incelenir.

5-Uygulamanın Doğrulanması: Uygulamanın süreç tanımına uygun yürütüldüğünün doğrulanması(gözden geçirilmesi, denetleme)



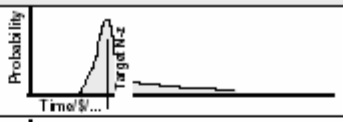
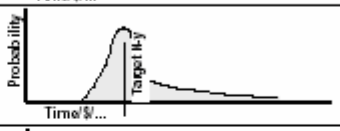
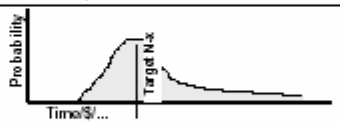
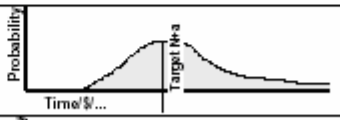
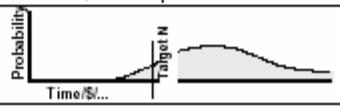
CMM de basamaklı bir yapı varken SPICE da ise böyle bir yapı yoktur.

3.6.3 CMM ve proje tahmin süreçleri arasındaki ilişki



Şekil 3.6 CMM Proje Tahmin Süreleri İlişkisi

CMM'de firmaların aldıkları notlarla, proje tahmin süreleri açısından bir ilişki vardır. Buna göre firmaların not düzeyi artıkça, proje öncesi tahmin süresi ile proje sonu gerçekleşen süre arasındaki fark azalacaktır.

Level	Process Characteristics	Predicted Performance
5 Optimizing	Process improvement is institutionalized	
4 Managed	Product and process are quantitatively controlled	
3 Defined	Software engineering and management processes defined and integrated	
2 Repeatable	Project management system in place; performance is repeatable	
1 Initial	Process is informal and ad hoc; performance is unpredictable	

Şekil 3.7 CMM Tahmin Düzeyleri

SEI'den alınan yukarıdaki tabloda da seviyeler ile tahmin süreleri arasındaki ilişki verilmiştir.[37]

3.6.4 CMM Düzeyleri

1.İlk Düzey(Başlangıç Düzeyi) : Yazılım süreci, gelişi güzeldir. Bazen de kaotiktir. Başarı, kişisel çabalara bağlıdır. Süreçler tanımlanmışsa da uyulmaz. Kriz durumunda yapılan plan terk edilir. Stabilite yoktur. Her proje farklı yapılır. Lider önemli ve etkindir. Yine de iyi projeler çıkartabilir. En çok yapılan iş kodlama ve testtir.

2. Tekrarlanılabilir Düzey: Temel proje usulleri konulmuştur. Zaman planı, maliyet, insan gücü, işlevsellik planlanır ve izlenir. Ürün dayanakları oluşturulmaktadır. Bu ürün dayanaklarına baseline adı verilir.

3. Tanımlı Düzey: Süreçler organizasyon çapında tanımlanmıştır ve belgelenmiştir. Eğitim verilmektedir. Yazılım geliştirme, yazılım yönetimi ve kurum yönetim süreçleri entegredir. Her projedeki standart sürçler o projeye uygulanır. SEPG (System Engineer Process Group)gurubu vardır.

4. Yönetilen Düzey: Yazılım sürecinin ve ürün kalitesinin ayrıntılı sayısal ölçümleri toplanır. Süreçler nicel olarak anlaşılmıştır.

5.En iyileşen Düzey: Süreç uygulamalarından gelen nicel bilgiler, yeni teknoloji ve buluşlarla yapılan pilot uygulamalarda sürekli iyileşme vardır. Organizasyon sürekli iyileşmeye odaklıdır. Süreç değişikliği ve teknoloji değişikliği ile iyileşme sürekli olur.

Tabi ki bu işlerin yapılması için yani düzeylerde artış olabilmesi için zamana ihtiyaç duyulmaktadır .”Olgunlaşmak Zaman İster”

3.6.5 CMM in Türkiye deki Yeri

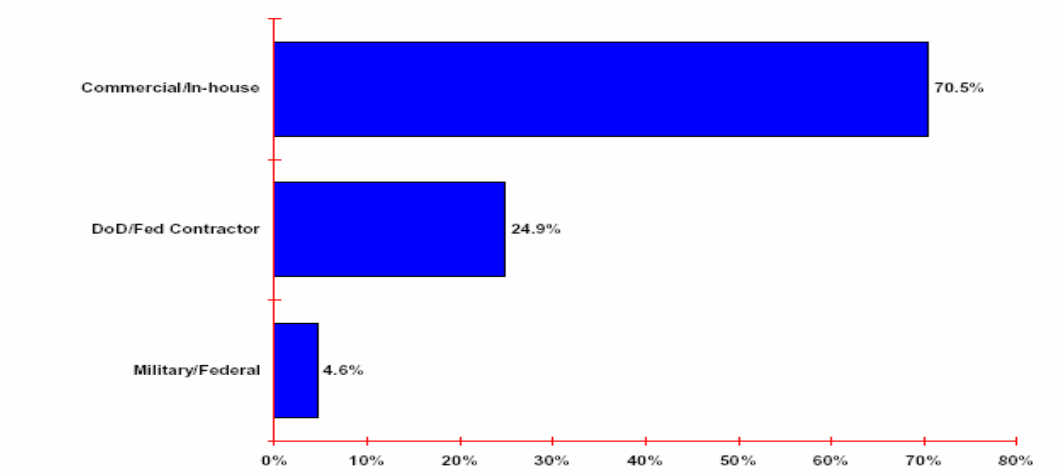
CMM‘in Türkiye’deki durumuna bakacak olursak 2 firmanın öne çıktığını görüyoruz. Bunlardan birincisi entegre yetenek olgunluk modeli seviye 5(CMMI-5) notu bulunan ve aynı zamanda ISO9001: 2000 NATO AQAP - 160 kalite belgeleri olan MilSOFT dur. İkinci olarak ta Vestel ’in büyük bir oranda ortaklığı bulunduğu ve SEI SW-CMM Seviye 3 uyumlu yazılım geliştirme yeteneğine sahip olan Aydın Yazılım ve Elektronik Sanayi A.Ş‘dir. Dünyadaki durumla karşılaştırıldığında seviye 4 ve seviye 5 deki firmaların 2002 yılına göre listesi aşağıda verilmiştir. Tabloya baktığımızda çok iyi bir dururumda olduğumuz görünmesine rağmen Türkiye olarak son yıllarda yazılım kalitesi olarak hızlı bir yükselme içinde olduğumuzu söyleye biliriz.

Tablo 3.6 2002 yılına göre ülkelerin CMM'e göre 4 veya 5 notu olan firmaların sayıları

Ülke	Seviye 4	Seviye 5
Singapur	1	
Kanada		1
Fransa	1	
İsrail	1	
İrlanda	1	
Avusturalya	2	
Hindistan	27	50
Çin		2
Rusya		1
Amerika	39	20
Dünya	**72**	**74**

3.6.6 CMM'in kullanan şirketlerin dağılımı

Başlangıçta Amerika Savunma Bakanlığının ihtiyacı için geliştirilen CMM geçen süre içerisinde askeri olmayan şirketler tarafından da kabul edilmeye başlamıştır. Aşağıdaki şekilde de görüldüğü üzere CMM'ı kullanan şirketlerin 70,5% ticari şirketlerdir.



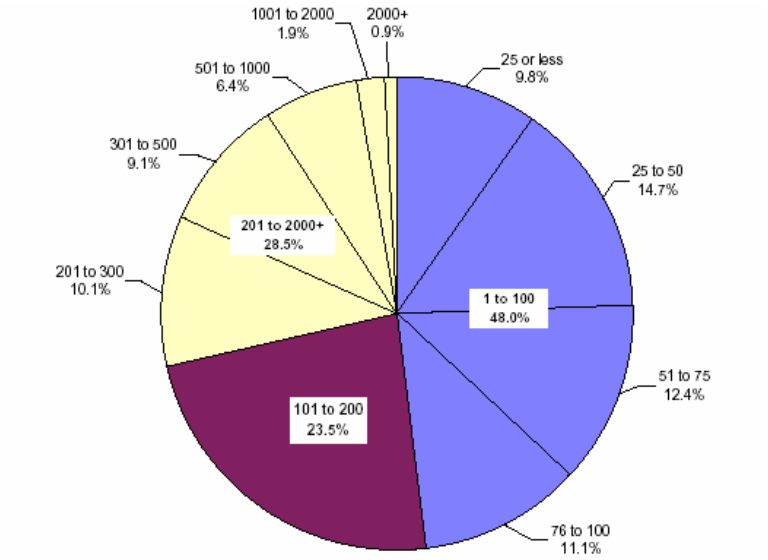
Şekil 3.8 CMM kullanan şirketlerin dağılımı

Yukarıda görülen bu tablo 2002 yılında CMM'i geliştiren SEI (Software Engineering Institute) tarafından oluşturulmuştur.

Firmaların CMM kullanım amaçlarına bakacak olursak 2 ana nokta üzerinde durulduğunu görüyoruz. Bu amaçlardan birincisi yeni açılan ihalelerde; ihalelere katılan firmaların yeni gelişen metodojileri takip etmelerinin yazılım firmalarından istenmesidir. İkinci amaç ise yazılım firmalarının kendi verimliliklerini arttırarak rakipleri ile aralarındaki rekabette öne çıkma istemeklerdir.

3.6.7 Yazılım mühendisleri ve CMM arasındaki ilişki:

Yazılım mühendisleri ile CMM kullanan firmaların arasındaki ilişkiye bakarsak SEI tarafından 1094 firma göz önüne alınarak hazırlanan rapora göre Firmaların personel sayılarının dağılımı aşağıda verilmiştir.



48%	1-100 kişi
23.5%	101-200 kişi
28.5%	201-200+ kişi

Şekil 3.9 CMM ve Personel sayıları

Buradan da gördüğümüz üzere yazılım geliştirme işinde genel itibari ile büyük yazılım geliştirme şirketleri tarafında CMM in tercih edildiğini görüyoruz.

CMM'da düzey artırımları için yapılması gerekenler listelenmiştir.

2.Düzeye Yükselmek İçin

- İsterler Yönetimi
- Yazılım Proje Planlaması
- Yazılım Proje İzleme ve Kontrol
- Yazılım Tasarım Yönetim
- Yazılım Kalite Güvencesi
- Yazılım Konfigürasyon Yönetimi

3.Düzeye Yükselmek İçin

- Kurum Süreç Odağı
- Kurum Süreç Tanımı, SEPG
- Tümleşik Yazılım Yönetimi
- Yazılım Ürün Mühendisliği
- Gruplar Arası Kordinasyon
- Eş düzey Gözden Geçirme

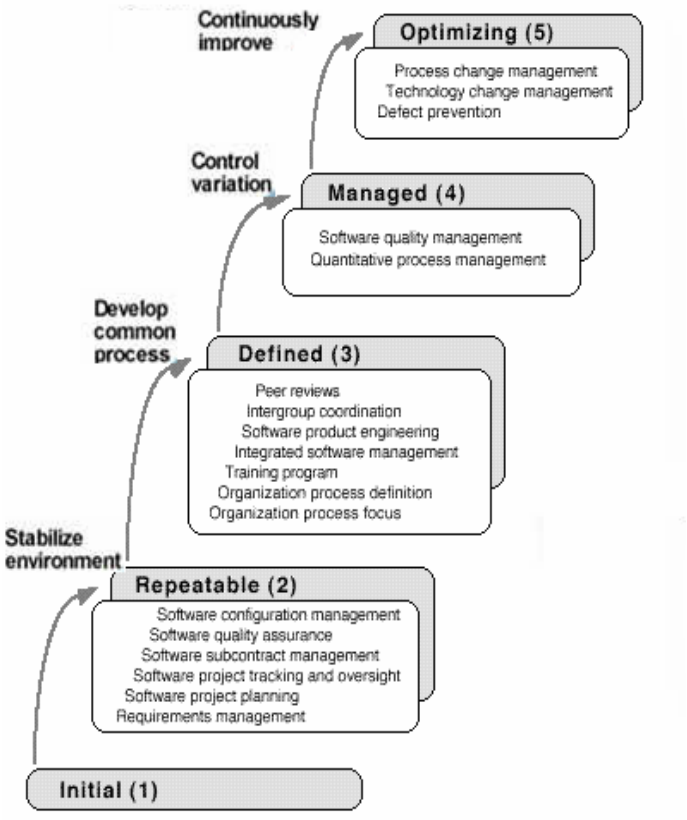
4.Düzeye Yükselmek İçin

- Nicel Süreç Yönetimi
- Yazılım Kalite Yönetimi

5.Düzeye Yükselmek İçin

- Hata Önleme
- Teknoloji Değişim Yönetimi
- Süreç Değişim Yönetimi

5.Düzeyde sayısal uygulama, sayısal model vardır. Mutlaka iyileşme yönünde değişim vardır.



Şekil 3.10 CMM ve Düzey Artırımları

3.6.8 CMM ile ilgili bilinmesi gerekenler

CMM yazılım isterlerinin ne şekilde karşılanacağını söylemez, işi yapmadan iş içerisindeki ihtiyaçların karşılanmasının zorunda olduğu üzerinde durur. Ayrıca sonradan ortaya çıkacak farklı ihtiyaçların da nasıl karşılanacağını önceden bilmemiz gerektiği konusunda bizi uyarır.
Bu ihtiyaçların karşılanması işini de iyi bir şekilde belgelenmesi gerektiğini söyler.

- Yazılımı gerçekleştirmedeki ihtiyaçları biliyor muyuz?
- İhtiyaçların proje sırasındaki değişmesi nasıl izleyeceğiz ve hangi işleri ne şekilde yapacağız?
- Yazılım mühendislerine proje içerisindeki görevleri tam olarak aktarıldı mı?
- Yazılımcılar proje içerisindeki süreçlere ne şekilde faydalı olduklarını biliyorlar mı?
- Yazılım mühendisleri işlerini yaptıkları zaman kendilerinden sonraki kısımda çalışanlara hangi bilgileri ulaştıracaklarından haberdarlar mı?

İhtiyaçları karşılamada asıl amaç projenin bütünüyle ilgili olan bir process oluşturmaktır. Bu sayede farklı gereksinimler bile olsa genel yapıda bir değişiklik olmayacaktır. CMMI ile ilgili bilgiler 4.Bölümde detaylı olarak anlatılacaktır.

3.7 ISO/IEC 15504 (SPICE[Software Process Improvment and Capability dEtermination])

SPCIE, Avrupa'nın yazılım geliştirme modeli arayışları içerisinde çerçevesinde oluşturulmuştur.

SPICE'ın çıkışında CMM e olan eleştirilerinde rolü büyük olmuştur.

Bu eleştiriler;Yalnız büyük projeler için uygundur.Gerekli gereksiz tek yol vardır Süreçlerin önemi az dikkate alınmıştır.Yeterince olgunluk ile yetenek arasındaki bilgileri ölçmüyordur.Gelişmeyi yeterince ölçmüyordur.

Amacı: Yetenek ve olgunluğu belirleme,Süreç iyileştirme sağlama

Kapsamı: Yazılım satın alma, geliştirme, işletme, bakım, onarım, destek aşamalarını kapsar.

3.7.1 SPICE ve ilkeler(Pricipes):

- Ortak modeller üstü, uluslar arası, kapsamlı bir çerçeve model.
- Diğer modelleri destekleyen uyumlu bir modeldir.
- Her endüstriye, organizasyona, duruma uygun modeldir.
- İyileşmeyi ve gelişmeyi ölçebilen.
- Nesnel tutarlı tekrarlanabilir.
- Sertifikasyon amacı taşımayan bir modeldir.

SPICE içerisinde her değerlendirme için bir sponsor vardır. Sponsor firma içinden veya dışından olabilir. SPICE değerlendirmesi bir ekip işidir. Ekip seçimi karışık olarak yapılıyorsa lider ve ağırlık dışarıdan olmalıdır.

Tablo 3.7 Spice ve Değerlendirme

		Ekip		
Sponsor		**İç**	**Dış**	**Karışık**
	İç	☒	☒	☒
	Dış	☐	☒	☒

3.7.2 SPICE belgeleri:

1.Kavramlar ve Giriş

2.Referans Modeli

3.Değerlendirme İstekleri

4.Değerlendirme Rehberi

5.Uyumlu Model

6.Değerlendirme Nitelikleri

7.Süreç İyileştirme Rehberi

8."Supplier" Yetenek Belirleme Rehberi

9.Sözlük

3.7.3 SPICE değerlendirme:

SPICE'da sonuç profil olarak verilir. Profil 2 boyutlu bir tablodur. Süreçlerin her birinin değerlendirilmesini sağlar. Süreçlere bir skala boyunca not verilmesi gerektir. Not düzeyi neyin başarılmış olduğuna bakar, nasıl bu işin başarıldığına bakmaz. Sürekli olarak iyileşmeyi ölçmeye yardımcı olmaya çalışır. Değerlendirmede bir referans modeli değil bir uyumlu model kullanır.

Uyumlu Model:

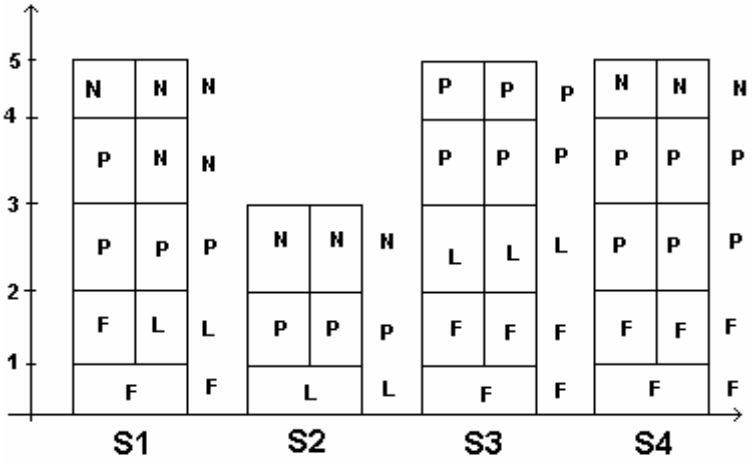
- En az bir süreci kapsamalı
- Sıfırdan başlayarak en az iki yetenek düzeyini içerisinde barındırmalı
- Referans modeline iz düşümü sağlanmalı
- Not dönüştürme mekanizması sağlanmalı

Yetenek Boyutu:

Tablo 3.8 Spice ve Yetenek Boyutu

	0	Eksik		0–5 arası bir not her süreç için verilir.
	1	İcra edilen		
	2	Yönetilen		
	3	Oturmuş		
	4	Kestirilebilen		
	5	En iyileşen		

Tablo 3.9 Spice ve Örnek Profil

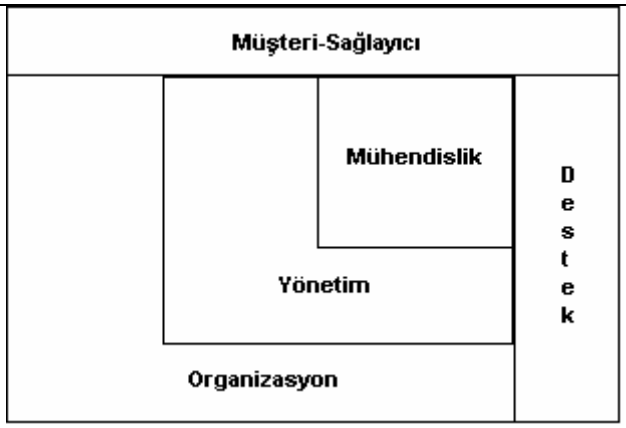
Süreç niteliği için notlar:	**Örnek Profil**
N: %0-%15 (none) P: %15-%50 (partly) L: %50-%85 (largely) F: %85 üssü başarı (fully)	

Notlar:S1 için 2 S2 için 1 S3 için 3 S4 için ise 2. Süreçlere verilen not en alt düzeyden başlar.”F” notu varsa yukarı çıkar. İlk “L” yi görünce durur ve o düzeyin notunu alır.”L” den önce “P” varsa bir alt düzeyin son “F” notunu alır.

3.7.4 SPICE süreç kategorileri:

SPICE süreç kategorileri 5 kısma ayrılmıştır.

Tablo 3.20 Spice ve Süreç Kategorileri

1. Müşteri-Sağlayıcı 2. Mühendislik 3. Yönetim 4. Organizasyon 5. Destek	

4. CMMI (Capability Maturity Model Integration)

CMMI yetenek olgunluk modeli, organizasyonların insan kaynaklarını, süreçlerini ve teknolojilerini organizasyonun iş yapabilme performansını uzun vadeli geliştirecek şekilde olgunlaştırmasını sağlayan bir modeldir. CMM modeli ile sağlanan başarıdan sonra üretilen SW-CMM, P-CMM gibi CMM ler ve sonucunda CMMI modeli oluşturulmuştur. CMMI modeli 1000 sayfadan daha fazla bilgi içermektedir. Temelinde organizasyonun faaliyetlerini geliştirme işi ile uğraşır. CMMI içerisinde 25 proje alanının her birinin belli amaçları vardır. Bu amaçlarında kendi içerisinde belli hedeflere ulaşılmasını ister.

Örnek olarak proje planlama süreci Establish Estimates, Develop a Project Plan gibi amaçlardan oluşur.[38]

Tablo 4.1 Proje Planlama Süreci

Project Planning	
Goal 1	Establish Estimates
Goal 2	Develop a Project Plan
Goal 3	Obtain Commitment to the Plan
Goal 4	Achieve Specific Goals
Goal 5	Institutionalize a Managed Process

Her bir amacın kendi içerisinde farklı aktiviteleri vardır

Goal 1 Establish Estimates:

- Estimate the Scope of the Project
- Establish Estimates of Work Product and Task Attributes
- Define Project Life Cycle
- Determine Estimates of Effort and Cost

Yukarıda görüldüğü gibi Establish Estimates kendi içerisinde 4 tane aktivitesi vardır. Proje Planlama süreci ise toplam 26 aktiviteden oluşur.

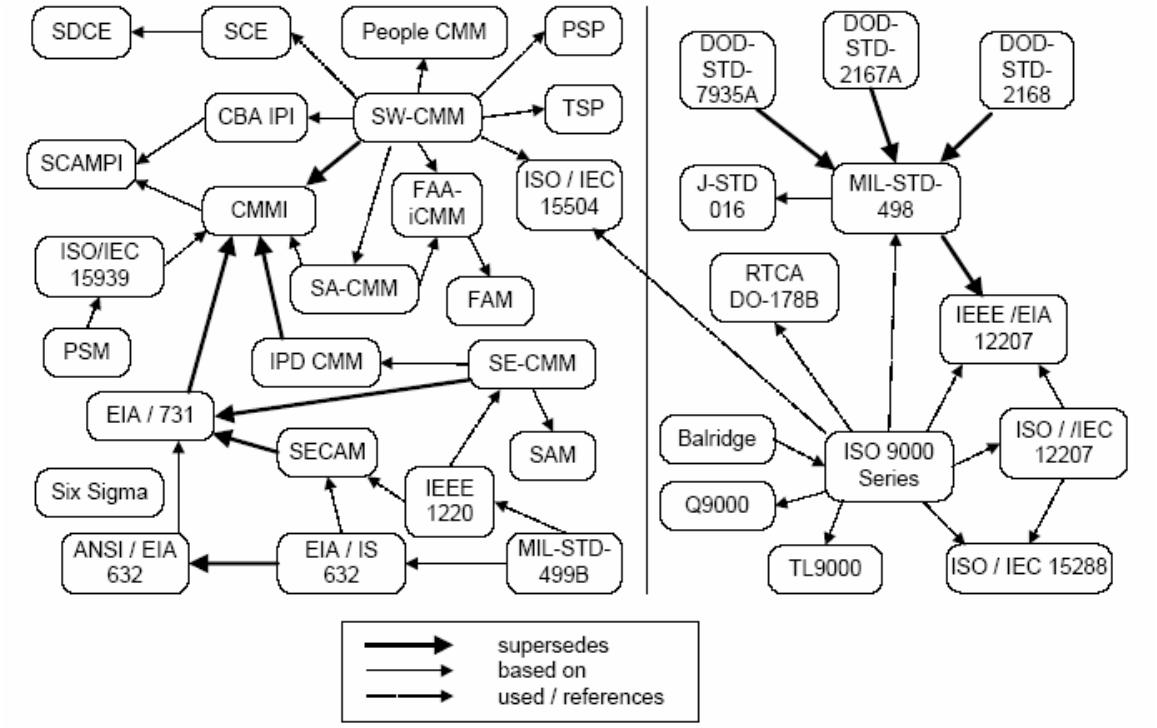
Organizasyon içerisinde süreçler gerçekleştirilirken tanımlanmış amaçlar doğrultusunda hareket edilmelidir.

4.1 CMMI Tarihsel Gelişimi

CMM&CMMI in tarihsel gelişimine bakmadan önce yazılım geliştirme sektörünün tarihsel gelişimine bakmak yararlı olacaktır.[39]

Yazılım Sektörü Tarihsel Gelişimi:

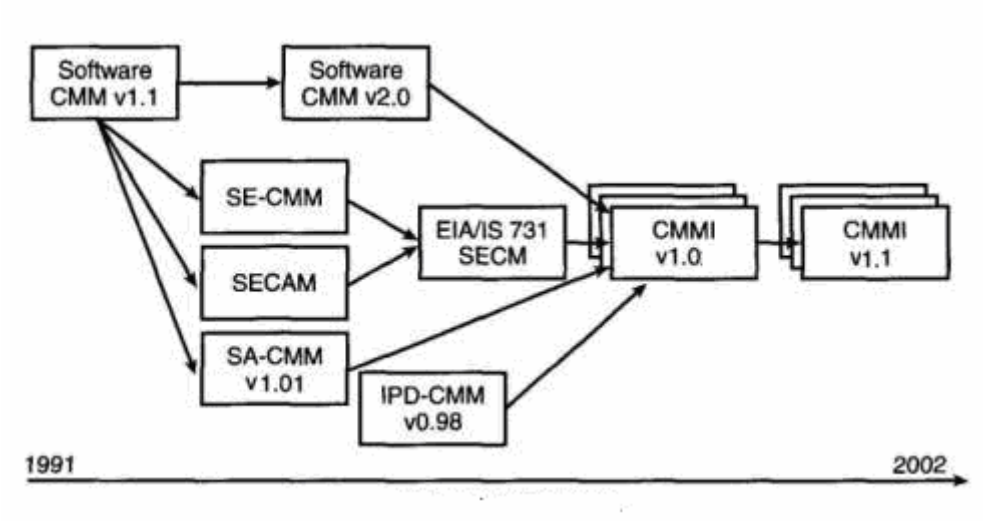
- 1924-İlk modern istatiksel kalite kontrol –Shewhart.
- 1940-Modern istatiksel kalite kullanılmaya başlıyor.
- 1946-Bilgisayarın doğumu.
- 1950-Juran, Deming ve Feigenbaum'un önderliğinde Japonya
- 1960-Kalite kontrol ve yönetim felsefesi(Japonların ulusunun en önemli işi olmalı!)
- 1968-NATO yazılım mühendisliği konferansı.
- 1970-Yazılım kalite terimi ilk kez kullanılıyor.
- 1980-Toplam kalite çalışmaları yapılmaya başlanıyor.



Şekil 4.1 CMMI Tarihsel Gelişimi

Tablo 4.2 CMMI Tarihsel Gelişimi

1979-Crosby's maturity grid (Quality is Free) 1985-IBM maturity grid(Radice) 1987-SEI software process maturity framework 1988-SEI software process domains 1990-SEI Software CMM V0.2 1991-SEI Software CMM V1.0 1993-SEI Software CMM V1.1 1995-SPICE Baseline Practices Guide	1995-People CMM 1995-Systems Engineering CMM 1996-Software Acquisition CMM 1997-IPD CMM 1997-EIA 731(Systems Engineering Capability Model) 2000-SEI CMM Integration v1.0(CMMI) 2002-SEI CMM Integration v1.1[40]



Şekil 4.2 CMMI Tarihsel Gelişimi

1996 yılında CMM 2.0 versiyonu 1991 yılında ortaya çıkan CMM'in yerini aldı 1997 yılında IPD-CMM V0.98 versiyonu ortaya çıkartıldı 1998 yılında CMM v1.1 'ın gelişmiş modelleri olan SE-CMM SECAM VE SA-CMMV1.01 birleşerek EIA/IS 731 SECM'i oluşturdu oluşan bu modeller 1999 yılının sonunda CMMI sürüm 1.0 'ı ortaya çıkardı daha sonrasında geliştirilen CMMI V1.0 2001'in sonunda CMMI V1.1.0 adını aldı.

4.2 CMMI-İlham Verenler:

CMMI ortaya çıkmadan önce bazı modeller oluşturulmuştur ve oluşturulan bu modeller CMMI oluşturulması sırasında esin kaynağı olarak kullanılmıştır.

- Crosby'nin olgunluk skalası[41]
- IBM'in süreç skalası

Bu ortaya konulan modeller CMMI'ın da temellerini oluşturmuştur.

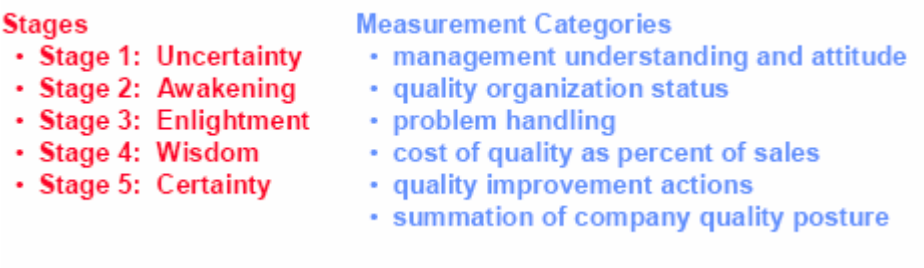
4.2.1 IBM olgunluk skalası:

- Geleneksel
- Uyarlama
- Bilgi
- Yetkinlik & Bilgelik
- Bütünleşik Yönetim Sistemi

4.2.2 CROSBY olgunluk skalası:

Kalite, uygunsuzluğunun neden olduğu maliyet olarak ölçülür.(yanlışların neden olduğu maliyet)[42]
Seviyeler:

- Seviye 1:Belirsizlik
- Seviye 2:Uyanma
- Seviye 3:Aydınlanma
- Seviye 4:Bilgelik
- Seviye 5:Emin Olma



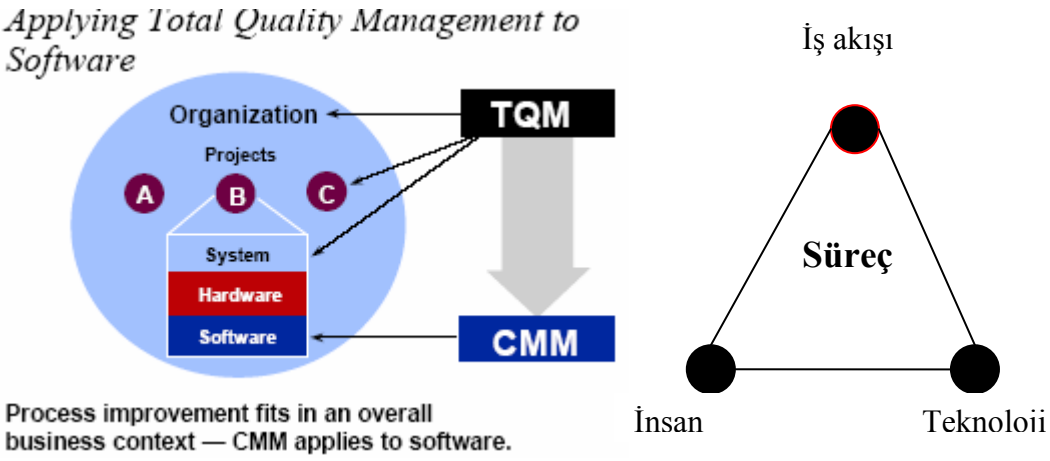
Tablo 4.3 Crosby'nin Olgunulk Skalası

	Belirsizlik	**Karşılaştırma**	**Uyanma**	**Aydınlanma**	**Kesinlik**
Yönetim Kavramları	Sertifike Olalım	Ödül alalım	Daha iyi Olamaz mıyız	Kalite hakkında ciddiyet	İşimizi daha iyi yapmamamız için bir neden yok
Sistem	Ödül Kredileri	Guru kasetleri almak ve göstermek	ISO 9000, Mil-Q-9858	"Gerçekten neyi bilmemiz gerekli?"	Önlem alma
Performans Standartları	İş yoğunluğu ile İlgili	Abul edilebilir kalite seviyeleri	Devamlı iyileşme	6 Sigma	Sıfır hata
Ölçümler	Fikirler	Karşılaştırma	Müşteri şikayetleri	Tam karşılaştırmalı ölçümler	Kalite maliyetleri
Kalite Bölümünün Durumu	Kalite,üretim ya da mühendislik bölümlerinin içinde gizli	Güçlü bir kalite lideri atanır ama hala ilk önceki ürünü teslim etmek üzerinedir.	Kalite bölmümü üst yönetime raporlar, kalite yöneticisi şirket yönetiminde yer alır.	Kalite bölümü şirketin yöneticilerindendir. Durum raporlaması ve engelleyici çalışmalar etkin bir biçimde yapılır.	Kalite yöneticisi
Kalite maliyetinin toplam satışa oranı	Raporlanan: Bilinmiyor Gerçekleşen: %20	Raporlanan: %3 Gerçekleşen: %18	Raporlanan: %8 Gerçekleşen: %12	Raporlanan: %6,5 Gerçekleşen: %8	Raporlanan: %2,5 Gerçekleşen: %2,5
Şirketin Kalite Pozisyonun un Durumu	"Neden kalite sorunlarımız olduğunu bilmiyoruz"	"Kalite sorunları kaçınamazmıyız?"	Yönetimin Kararlılığı ve kalite iyleştirme alışmaları ile sorunlarımızı tespit ediyoruz ve çözüyoruz"	"Hata önleme bizim günlük işimizin bir parçasıdır."	"Neden kalite sorunlarımızın olmadığını bilmiyoruz"

4.3 CMMI-TKY

1950'lerde başlayan TQM(Total Quality Management) alanındaki çalışmalar, başarılı neticelerde alınca, akıllara yazılım geliştirme işi içinde uygulanabilir olup olmayacağı sorusu gelmiştir.

Bu toplam kalite çalışmalarının yazılım için nasıl uygulana bileceği sorusu üzerine odaklanılmış ve sonucunda yazılım için özel olarak geliştirilen CMMI oluşturulmuştur.[43]

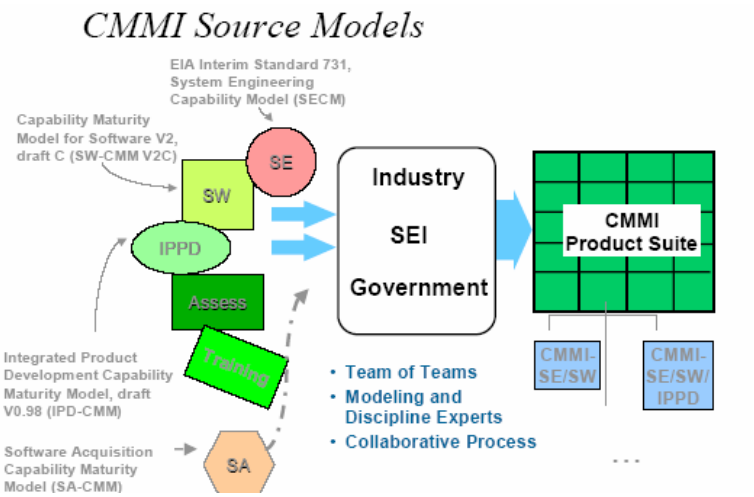


Şekil 4.3 CMMI & TQM ilişkisi

CMMI'a göre süreç, sadece iş akışı değil, insan ve teknoloji desteğini de içermelidir.

4.4 CMMI Kaynak Model:

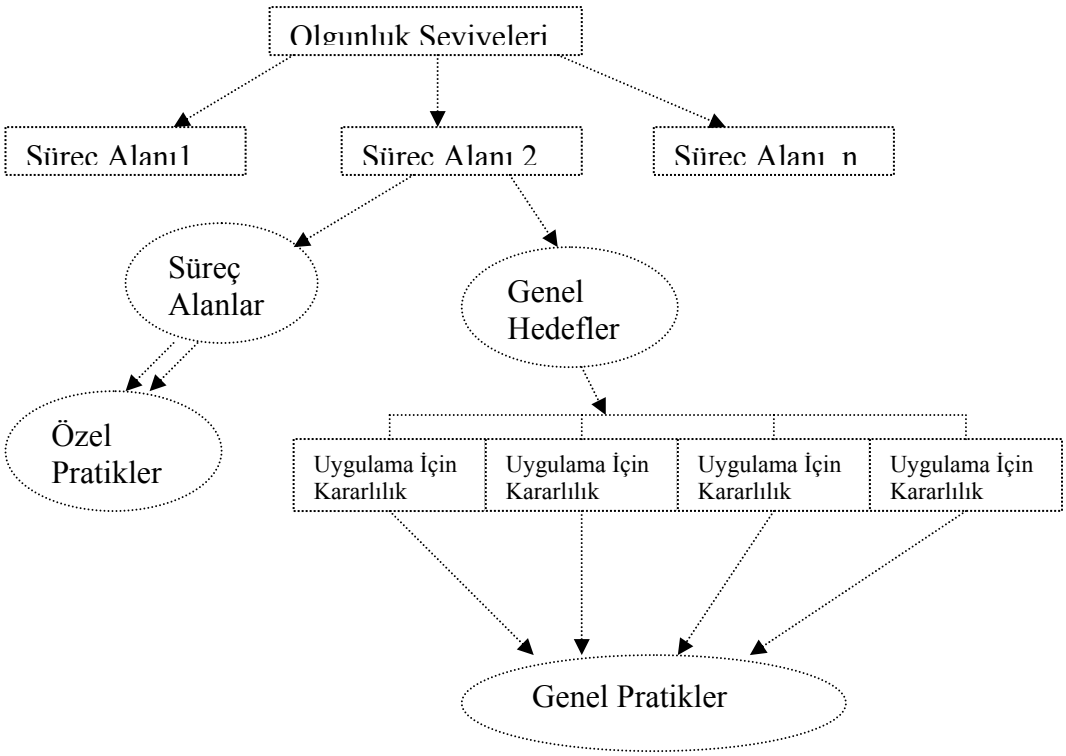
Yıllar içerisinde geliştirilen farklı modellerin bazı artı ve eksi yanları vardı. Bu modellerin her biri için farklı zamanlarda farklı eğitimler alınması gerekiyordu. Alınan bu eğitimlerin maliyeti ve iş akışlarını olumsuz yönde etkiliyordu. Geliştirilen bu modelleri bir arada toplayarak nasıl daha iyi bir model ortaya çıkarılır sorusu üzerinde durulması sonucunda işleri farklı CMM'ler de yapılmasındansa tek bir model üzerinde yapılması fikri doğdu.[44]



Şekil 4.5 CMMI Kaynak Modeli

4.5 CMMI- Temel Yapısı

CMMI ın basamaklı modeli aşağıda görülmektedir.[45]



Şekil 4.4 CMMIBasamaklı Modelin Yapısı

Yanlış Bilgi: Yazılım Sorunları teknik sorunlardır.

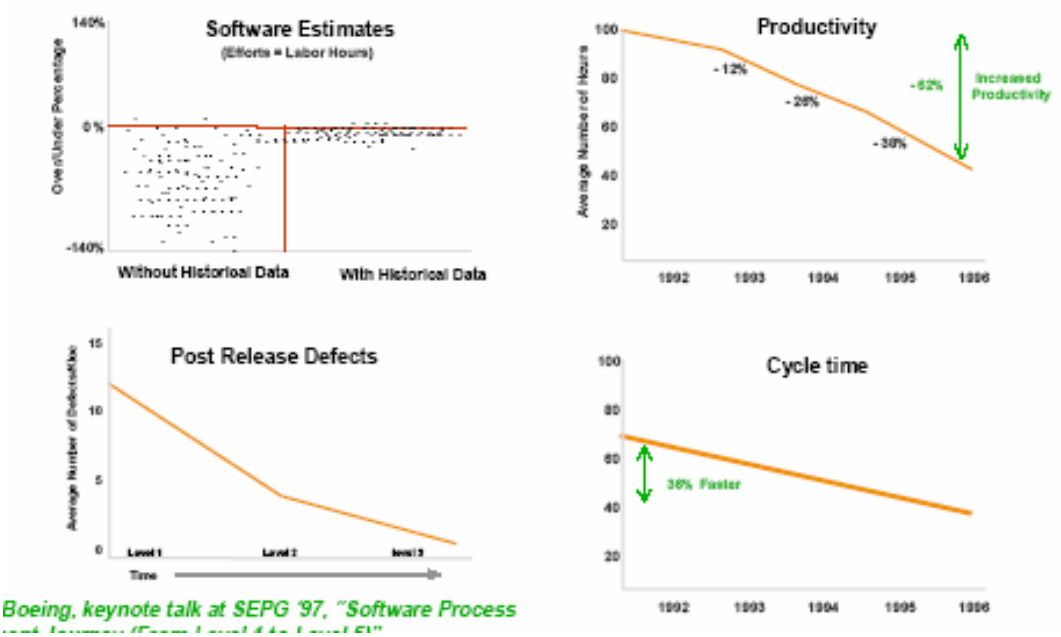
Doğru Bilgi: Yazılım gelişmesinde esas sorunlar yönetseldir. Teknik değildir.

CMMI, şirketler için yazılım geliştirmedeki durumlarını görmelerine yarayan bir

check-up tır. Şirketler bu sayede hangi alanlarda ne gibi eksiklikleri var bunları görebilirler. CMMI sorunların neler olduğunu söyler bunların nasıl çözüleceğini söylemez.

4.6 CMMI Faydaları

- Yazılımın belirlenen zaman diliminde bitmesine yardımcı olur.
- Daha kısa sürede ürün gelişimine olanak sağlar.
- Üretim maliyetlerinde azalma sağlar.
- Yazılım hatalarının sayısını azaltır.



[46] **Şekil 4.6** CMMI Faydaları

CMMI kullanıp ta başarılı olan şirketlerin iyileşme oranları [47]

Tablo 4.4 CMMI Kullanımında Başarı Oranları

% iyileşme	Medyan	En Küçük	En Büyük
Yıllık üretkenlik kananımı	35	9	67
Hataların erken teşhisinden kaynaklanan kazanımlar	22	6	25
Proje Süresinin Kısalması	19	15	23
Hizmete giriş sonrası hatalardan sağlanan kazanımlar	39	10	94
Yatırım geri dönüşü(ROI)	500	420	880

4.7 CMMI-2002 Seviyeleri [48]

Tablo 4.5 CMMI Seviyeleri

2.Seviye	**3.Seviye**	**4.Seviye**	**5.Seviye**
Yönetilen	**Tanımlı**	**SayılarlaYönetilen**	**Optimum**
ProjeYönetimi	**Mühendislik ve Destek**	**Ürün ve Süreç Kalitesi**	**Sürekli İyileşme**
1.Müşteri İsterleriYönetimi	1.Müşteri ve İsterleri Geliştirme	1.Kurumsal Süreç Performansı	1.Kurumsal Yaratıcılık ve Yaygınlaştırma
2.Proje Planlama	2.Teknik çözüm	2. Sayısal Proje Yönetimi	2. Sebep Analiz ve Çözüm
3.Proje İzleme ve Takip	3.Kontrol (Verifation)		
4.Tedarikçi Sözleşme Yönetimi	4.Doğrulama (Validation)		
5.Ürün ve Süreç Kalite Güvencesi	5.Ürün Bütünleştirme		
6.Konfigürasyon Yönetimi	6.Kurumsal Süreç Odaklanması		
7.Ölçme ve Değerlendirme	7.Kurumsal Süreç Tanımlama		
	8.Kurumsal Eğitim		
	9.Bütünleşik Proje Yönetimi		
	10.Bütünleşik Takım Olşturma		
	11. Bütünleşme için Orgazizasyonel Ortam		
	12. Bütünleşik Tedarikçi Yönetimi		

	13. Risk Yönetimi		
	14. Karar Analiz ve ÇözümÜretme		

4.7.1 CMMI-2002

Tablo 4.6 CMMI Süreçleri

Seviye	PY	SY	D	Müh	Süreç Alanı	Ö H S	ÖU	G H	GP [1]
2				Tml	Gereksinim Yönetimi	1	5	2	12
2	Tml				Proje Planlama	3	14	2	12
2	Tml				Proje İzleme ve Takip	2	10	2	12
2	Tml				Tedarikçi Sözleşme Yönetimi	2	7	2	12
2			Tml		Konfigürasyon Yönetimi	3	7	2	12
2			Tml		Ürün ve Süreç Kalite Güvencesi	2	4	2	12
2			Tml		Ölçme ve Değerlendireme	2	8	2	12
3				Tml	Gereksinim Geliştirme	3	10	2	12
3				Tml	Teknik Çözüm	3	9	2	12
3				Tml	Sağlama	3	8	2	12
3				Tml	Doğrulama	2	5	2	12
3				Tml	Ürün Bütünleştirme	3	9	2	12
3		Tml			Kurumsal Süreç Odaklanması	2	7	2	12
3		Tml			Kurumsal Süreç Tanımlama	1	5	2	12
3		Tml			Kurumsal Eğitim	2	7	2	12
3	İleri				Bütünleşik Proje Yönetimi	4	13	2	12
3	İleri				Risk Yönetimi	3	7	2	12
3			İleri		Karar Analiz ve Çözüm	1	6	2	12

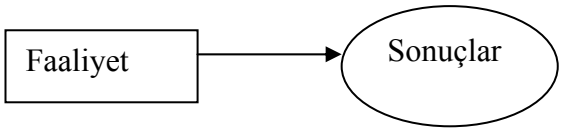
[1] Kısaltmaların Açılımları: [49]
PY:Proje Yönetimi , SY:Sayılarla Yönetilen , D:Destek , Müh:Mühendislik , ÖHS:Özel Hedef Sayısı, ÖUÖzel Uygulama Sayısı ,GH:Genel Hedef, GP:Genel Pratikler , Tml:Temel

3	İleri				Üretme	2	8	2	12
3			İleri		Bütünleşik Takım Oluşturma	2	6	2	12
3	İleri				Bütünleşme İçin Organisasyonel Ortam	2	5	2	12
					Bütünleşik Tedarikçi Yönetimi				
4	İleri				Sayısal Proje Yönetimi	2	8	2	12
4		İleri			Kurumsal Süreç Performansı	1	5	2	12
5		İleri			Kurumsal Yaratıcılık ve Yayma	2	7	2	12
5			İleri		Sebep analız ve Çözüm Üretme	2	5	2	12
					Toplam	55	185	50	300

4.8 CMMI Süreçleri

- Süreç iyileştirme çalışmalarının en az geri dönüşü dünya ortalamasında yaklaşık olarak 1'e 4 dür[50]

4.8.1CMMI seviye 1

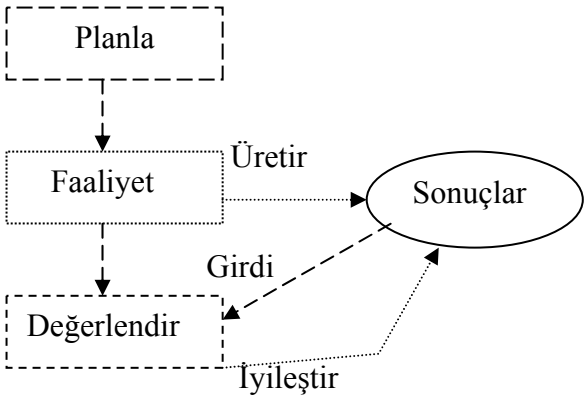


Şekil 4.7 CMMI Seviye 1

Seviye 1: Sadece Yap –Nasıl Yaparsan Yap!(Just do it.)

4.8.2 CMMI seviye 2

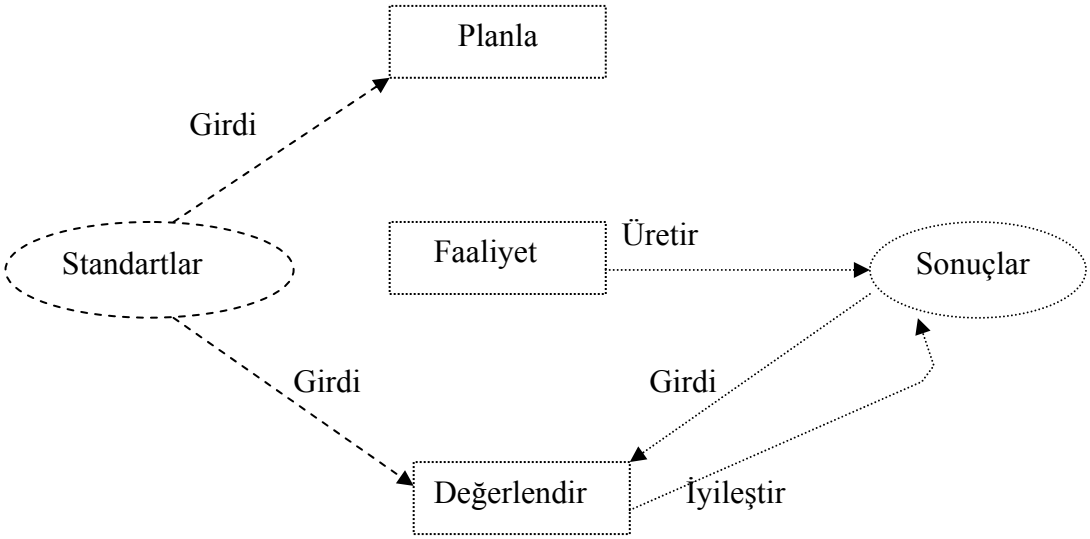
Doğru şeyi yaptığından emin olmak için, harekete geçmeden düşünülür ve faaliyet sonrası değerlendirme yapılır. Temel testler ve mevcut durum için bir plan vardır.



Şekil 4.8 CMMI Seviye 2

4.8.3 CMMI seviye 3:

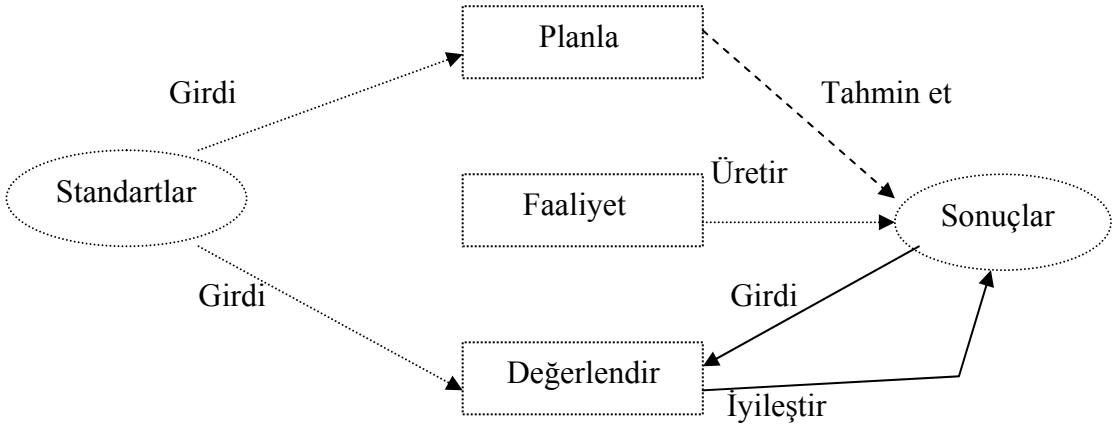
Hatalardan alınan dersleri kullanılır, standartlar oluştur. Plan değerlendirilir. Plan içerisindeki görevler belirlenir. Yaptığımız testler bir standarda bağlanır.



Şekil 4.9 CMMI Seviye 3

4.8.4 CMMI seviye 4

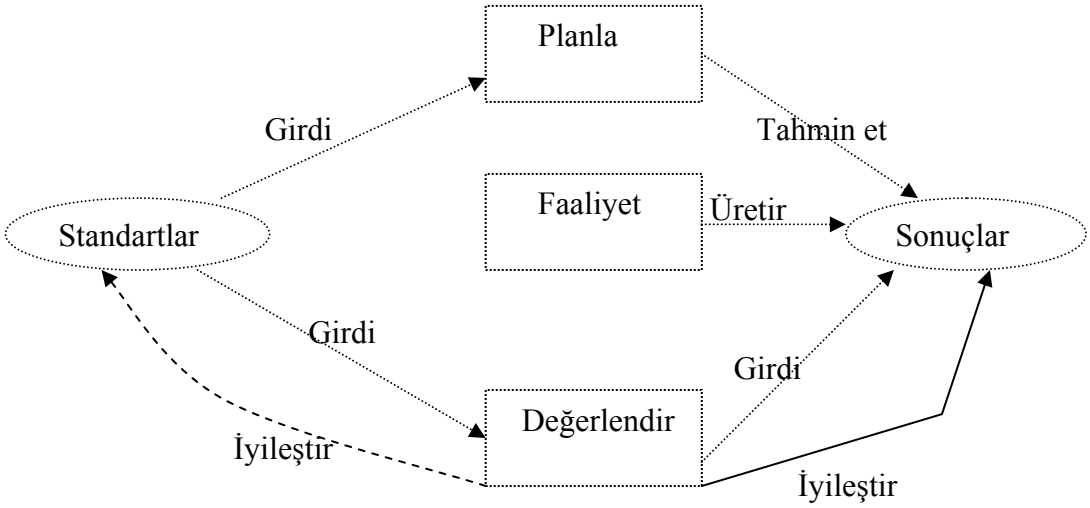
Sonuçları daha önceden tahmin et ve tahmin edildiği gibi sonuçlanması için emek harca. Ortak bir proje havuzu kullan. Sahip olduğumuzdan bilgi birikimine göre yapılan projelerle şimdiki yaptığımız projeleri kıyaslayarak ölçüp onlardan sonuç çıkart. Çıkartılan proje sonuçlarını değerlendir. Testlerden alınan sonuçların doğruluğuna bak.[51]



Şekil 4.10 CMMI Seviye 4

4.8.5 CMMI seviye 5

Hatalardan alınan dersleri, devamlı ve sürekli kullan. Standartları sürekli iyileştir. Sürekli olarak yapılanları daha ucuza ve daha az zamanda yap.



Şekil 4.11 CMMI Seviye 5

4.9 CMMI 2 Seviye Süreçleri

OS(Olgunluk Süreci)2.1-Gereksinim Yönetimi

Gereksinim Yönetiminin amacı, proje ürünleri ve ürün bilişenleri ile ilgili gereksinimleri yönetmek ve bu gereksinimler ile proje ürünleri ve ürün bileşenleri arasındaki tutarsızlıkları tespit etmektir.[52]

ÖH(Özel Hedef)1:Gereksinimlerin Yönetilmesi

- ÖU(Özel Uygulama)1.1: Gereksinimler üzerinde ortak bir anlayış geliştirilmesi
- ÖU(Özel Uygulama)1.2: Gereksinimler için taahhütlerin alınması
- ÖU(Özel Uygulama)1.3:Gereksinimlerdeki değişimlerin yönetilmesi
- ÖU(Özel Uygulama)1.4:Gereksinimler üzerinde çift taraflı takip edilebilirliğin sağlanması

- ÖU(Özel Uygulama)1.5:Proje ürünleri ile gereksinimler arasındaki tutarsızlıkların saptanması

GH(Genel Hedef)2:Yönetilir bir sürecin kurumsallaştırılması

GH(Genel Hedef)3:Tanımlı bir sürecin kurumsallaştırılması

OS2.2-Proje Planlama

Proje Planlamanın amacı, proje faaliyetlerini tanımlayan planları oluşturmak ve güncel tutmaktır.

Temel Faaliyetler

- Proje Planını oluşturmak
- Paydaşlarla gerekli iletişimin kurulması
- Plana taahhütlerin alınması
- Planı güncel tutmak

ÖH1:Kestirimlerin Oluşturulması

- ÖU1.1:Proje kapsamının belirlenmesi
- ÖU.12:İş ürünleri ve görev özellikleri için kestirimlerin oluşturulması
- ÖU1.3:Proje hayat döngüsünün tanımlanması
- ÖU1.4:Emek ve gider hesaplarının yapılması

ÖH2:Proje planının geliştirilmesi

- ÖU2.1:Bütçe ve zaman çizelgesinin oluşturulması
- ÖU2.2:Proje risklerinin saptanması
- ÖU2.3:Veri yönteminin planlanması
- ÖU2.4:Proje kaynaklarının planlanması
- ÖU2.5:Gerekli bilgi ve yeteneklerin saptanması
- ÖU2.6:Paydaşların ve katılımların planlanması
- ÖU2.7:Proje planının oluşturulması

ÖH3:Plana olan taahhütlerin oluşturulması

- ÖU3.1:Projeden etkilenen proje planlarının gözden geçirilmesi

- ÖU3.2:İş ve kaynak seviyelerinin ayarlanması
- ÖU3.3:Proje taahhütlerinin oluşturulması

GH2:Yönetilir bir sürecin kurumsallaştırılması
GH3:Tanımlı bir sürecin kurumsallaştırılması

OS2.3 Proje İzleme ve Takip

Proje izleme ve takibin amacı, projenin gidişatı hakkında sürekli bilgi toplanması ve böylece projenin performansı planlardan önemli ölçüde sapar ise bu durumun bir an önce fark edilmesi ve düzeltici aksiyonların alınabilmesidir.
ÖH1:Proje plana göre takip edilmesi

- ÖU1.1:Proje planlama parametrelerinin takip edilmesi
- ÖU1.2:Taahhütlerin takip edilmesi
- ÖU1.3:Proje risklerinin takip edilmesi
- ÖU1.4:Veri yönetiminin takip edilmesi
- ÖU1.5:Proje paydaşlarının katılımlarının takip edilmesi
- ÖU1.6:Gelişme toplantısı yapılması
- ÖU1.7:Kilometre taşı toplantılarının yapılması

ÖH2:Düzeltici aksiyonların kapanıncaya kadar takip edilmesi

ÖU2.1:Sorunların analizi
ÖU2.2:Düzeltici aksiyonların tespit edilmesi
ÖU2.3:Düzeltici Aksiyonların Yönetilmesi

GH2:Yönetilir bir sürecin kurumsallaştırılması
GH3:Tanımlı bir sürecin kurumsallaştırılması[52]

OS2.4 Tedarikçi Sözleşme Yönetimi

Tedarikçi sözleşme yönetiminin amacı, tedarikçiden ürün satın işlemini taraflar arasında imzalanmış bir sözleşme ile yönetmektir.
<u>Temel Faaliyetler</u>
Satın alma tipinin belirlenmesi

Sözleşmenin oluşturulması ve güncel tutulması

Sözleşmenin yerine getirilmesi

Satın alınan ürünlerin teslim alınması(kabul testi)

Satın alınan ürünün kullanıma alınması(proje aktarılması)

ÖH1:Tedarikçi Sözleşmesinin oluşturulması

- ÖU1.1:Satın alma tipinin belirlenmesi
- ÖU1.2:Tedarikçinin seçilmesi
- ÖU1.3:Tedarikçi Sözleşmesinin oluşturulması

ÖH2:Tedarikçi Sözleşmesinin gerekenlerinin yerine getirilmesi

- ÖU2.1:Ticari(COTS: Commercial of the Shell) ürünlerin incelenmesi
- ÖU2.2:Tedarikçi sözleşmesinin uygulanması
- ÖU2.3:Satın alınan ürünün kabul edilmesi
- ÖU2.4:Ürünün kullanıma geçiş süreci

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaştırılması

OS2.5 Konfigürasyon Yönetimi

Konfigürasyon yönetiminin amacı, konfigürasyon tanımlama, konfigürasyon kontrol, konfigürasyon durum değerlendirme ve konfigürasyon denetleme faaliyetlerini kullanarak iş ürünlerinin tutarlılığını ve bütünlüğünü oluşturmak ve korumaktır.

ÖH1:Temel sürümlerin oluşturulması

- ÖU1.1:Konfigürasyon parçalarının belirlenmesi
- ÖU1.2:Konfigürasyon Yönetim Sisteminin oluşturulması
- ÖU1.3:Temel sürümlerin yaratılması ve sürümlerin(versiyon) takip edilmesi

ÖH2:Değişikliklerin takip ve kontrol edilmesi

- ÖU2.1:Değişiklik isteklerinin takibi
- ÖU2.2:Konfigürasyon parçalarının kontrol edilmesi

ÖH3:Bütünlüğün sağlanması

- ÖU3.1:Konfigürasyon yönetim kayıtlarının oluşturulması
- ÖU3.2: Konfigürasyon denetimlerinin oluşturulması

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaştırılması

OS2.6 Süreç ve Ürün Kalite Güvencesi

Süreç ve ürün kalite güvencesinin amacı, çalışanlara ve yöneticilere, süreçler ve ilgili iş ürünleri ile ilgili,tarafsız gözlemler sunmaktır.[53]

Temel Faaliyetler

Gerçekleşen süreçlerin, iş ürünlerinin ve hizmetlerin, süreç tanımları, standartlar ve prosedürler ile tarafsız olarak karşılaştırmak

Uygunsuzlukları tespit etmek ve belgelenmek

Proje ekibine ve yönetime kalite güvence aktivitelerinin sonuçları hakkında bilgi vermek

Uygunsuzlukların üzerine gidildiğinden emin olmak

ÖH1:Süreçlerin ve iş ürünlerinin tarafsız olarak değerlendirilmesi

- ÖU1.1:Süreçlerin tarafsızca değerlendirilmesi
- ÖU1.2:İş ürünleri ve hizmetlerin tarafsız olarak değerlendirilmesi

ÖH2:Tarafsız olarak veri toplanması

- ÖU2.1:Uygunsuzlukların ilan edilmesi ve çözünüldüğünden emin olunması
- ÖU2.2:Kayıtların oluşturulması

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaştırılması

OS2.7 Ölçme ve Analiz

Ölçme ve analizin amacı, yönetimin bilgi ihtiyacına destek olmak için uygun ölçme yeteneklerini geliştirmek ve devamlılığını sağlamaktır.

ÖH1: Ölçme ve analiz çalışmaları için uygun altyapının oluşturulması

- ÖU1.1 Ölçme hedefinin belirlenmesi
- ÖU1.2: Ölçütlerin saptanması
- ÖU1.3:Veri toplama ve saklama prosedürlerinin belirlenmesi
- ÖU1.4:Analiz prosedürlerinin belirlenmesi

ÖH2:Ölçme sonuçlarının ilan edilmesi

- ÖU2.1:Ölçme verilerinin toplanması
- ÖU2.2:Ölçme verilerinin analiz edilmesi
- ÖU2.3:Veri ve analiz sonuçlarının saklanması
- ÖU2.4:Sonuçların ilan edilmesi

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaştırılması

4.10 CMMI 3 Seviye Süreçleri

OS3.1 Müşteri İsterleri Geliştirme

Müşteri isterleri geliştirmenin amacı müşteri, ürün ve ürün bileşenlerinin isterlerini(gereksinimlerini) oluşturmak ve analiz etmektir.[54]

Temel Faaliyetler

Müşteri ihtiyaçlarının, beklentilerinin ve kısıtlarının alınması, analiz edilmesi, doğrulanması ve gerekli iletişimlerin sağlanması

Paydaşların ihtiyaçların toplanması ve koordine edilmesi

Ürünün hayat döngüsü ihtiyaçlarının oluşturulması

Müşteri isterlerinin oluşturulması

Müşteri isterleri ile tutarlı olarak ürün ve ürün bileşenlerinin ilk(başlangıç) isterlerinin oluşturulması

ÖH1:Müşteri gereksinimlerinin geliştirilmesi

- ÖU1.1:İhtiyaçların meydana çıkarılması
- ÖU1.2:Müşteri gereksinimlerinin geliştirilmesi

ÖH2:Ürün gereksinimlerinin geliştirilmesi

- ÖU2.1:Ürün bileşenlerinin gereksinimlerinin oluşturulması
- ÖU2.2:Gereksinimlerin ürün bileşenlerine dağıtılması
- ÖU2.3:Arayüz gereksinimlerinin belirlenmesi

ÖH3:Gereksinimlerin analiz edilmesi ve doğrulanması

- ÖU3.1:Kullanıcı senaryoları ve kavramlarının oluşturulması
- ÖU3.2:Gereksinimlerin duyulan fonksiyonlar için tanımlar oluşturulması
- ÖU3.3: Gereksinimlerin analiz edilmesi
- ÖU3.4:Dengenin sağlanması için gereksinimlerin analizi
- ÖU3.5:Ayrıntılı yöntemler ile gereksinimlerin doğrulanması

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaştırılması

OS3.2 Teknik Çözüm

Teknik çözümüm amacı, müşteri isterlerine uygun çözümler tasarlamak, oluşturmak ve üretmektir. Çözümler, ürünleri, ürün bileşenleri ve ürün hayat döngüsü ile ilgili süreçlerin tamamını kapsar.[55]

Temel faaliyetler

Müşteri isterlerini sağlayan olası çözümlerin değerlendirilmesi ve en uygun çözümün seçilmesi.Seçilen çözüm için detaylı tasarımların oluşturulması.Tasarımların ürün ve ürün bileşenleri olarak üretilmesi

ÖH1:Ürün bileşenleri için çözümlerin seçilmesi

- ÖU1.1:Ayrıntılı alternatif çözümlerin geliştirilmesi ve seçim kriterlerinin belirlenmesi
- ÖU1.2:Kullanıcı senaryolarının ve kavramların ilerletilmesi
- ÖU1.3:Ürün bileşeni çözümlerinin seçilmesi

ÖH2:Tasarımın Geliştirilmesi

- ÖU2.1:Ürün ve ürün bileşenlerinin tasarımının yapılması
- ÖU2.2:Teknik veriler dosyası hazırlanması
- ÖU2.3:Kriterlere uygun olarak arayüzlerin tasarımlarının yapılması
- ÖU2.4:Yap\Al ya da tekrar kullanım analizlerinin yapılması

ÖH3:Ürün tasarımının hayata geçirilmesi

- ÖU3.1:Tasarımların gerçekleştirilmesi
- ÖU3.2:Ürün destek dokümantasyonun geliştirilmesi

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaştırılması

OS3.3 Sağlama

Sağlamanın amacı, seçilen iş ürünlerinin belirlenen isterleri karşıladığından emin olmaktır.[56]

Temel faaliyetler

Sağlaması yapılacak iş ürünlerinin seçilmesi

Sağlama için kullanılacak ortamın oluşturulması

Sağlama yöntem ve kriterlerinin belirlenmesi
Sağlamanın gerçekleştirilmesi

ÖH1:Sağlama için hazırlık yapılması

- ÖU1.1:Sağlaması yapılacak iş ürünlerinin belirlenmesi
- ÖU1.2:Sağlama ortamının oluşturulması
- ÖU1.3:Sağlama prosedürlerinin ve kriterlerinin oluşturulması

ÖH2:İkili gözden geçirmelerin gerçekleşmesi

- ÖU2.1:İkili gözden geçirme için hazırlıkların yapılması
- ÖU2.2: İkili gözden geçirmelerin gerçekleştirilmesi
- ÖU2.3: İkili gözden geçirme verilerinin analizinin yapılması

ÖH3:Seçilen iş ürünlerinin sağlamasının yapılması

- ÖU3.1: Sağlamasının yapılması
- ÖU3.2: Sağlama sonuçlarının analiz edilmesi ve düzeltici aksiyonların tespit edilmesi

GH2:Yönetilir bir sürecin kurumsallaştırılması
GH3:Tanımlı bir sürecin kurumsallaştırılması

OS3.4 Doğrulama

Doğrulamanın amacı, ürün ya da ürün bileşenlerinin hedeflenen ortama yerleştirildiğinde hedeflenen kullanımını sağladığının gösterilmesidir

Temel faaliyetler

Doğrulaması yapılacak iş ürünlerinin seçilmesi
Doğrulama için kullanılacak ortamın oluşturulması
Doğrulama yöntem ve kriterlerinin belirlenmesi
Doğrulamanın gerçekleştirilmesi

ÖH1:Doğrulama için hazırlıkların yapılması

- ÖU1.1:Doğrulanacak iş ürünlerinin seçilmesi
- ÖU1.2: Doğrulama ortamlarının oluşturulması
- ÖU1.3: Doğrulama prosedürlerinin ve kriterlerinin oluşturulması

ÖH2:Ürün ve ürün bileşenlerinin doğrulanması

- ÖU2.1 Doğrulamanın gerçekleştirilmesi

- ÖU:2.2: Doğrulama sonuçlarının analiz edilmesi

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3: Tanımlı bir sürecin kurumsallaştırılması

OS3.5 Ürün Bütünleştirme

Amacı, ürün bileşenlerini bir araya getirerek ürünü oluşturmak, oluşan ürünün isterlere uygun olarak çalıştığını görmek ve ürünü teslim etmektir.

ÖH1:Ürün bütünleştirme için hazırlıkların yapılması

- ÖU1.1:Bütünleştirme sırasının belirlenmesi
- ÖU1.2:Ürün bileştirme ortamının belirlenmesi
- ÖU1.3:Ürün bileştirme prosedürlerinin ve kriterlerinin belirlenmesi

ÖU2: Arayüz uyumundan emin olma

- ÖU2.1:Arayüz tanımlarının eksiksiz olduğunun gözden geçirilmesi
- ÖU2.2:Arayüzlerin yönetilmesi

ÖH3:Ürün bileşenlerinin bir araya getirilmesi ve ürün teslim edilmesi

- ÖU3.1:Ürün bileşenlerinin bütünleştirme için hazır olduğunun konfirme edilmesi
- ÖU3.2:Ürün bileşenlerinin bir araya getirilmesi
- ÖU3.3:Bir araya getirilen ürünlerin değerlendirilmesi
- ÖU3.4:Ürün ve ürün bileşenlerinin paketlenmesi ve teslim edilmesi

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaşması

OS 3.6Kurumsal Süreç Odaklanması

Amacı, kurumun süreç ve süreç varlıklarında hali hazırda bilinen güçlü ve zayıf yönlerine uygun olarak gerçekleştirilmesi gereken süreç iyileştirme çalışmalarını planlamak ve gerçekleştirmektir.

ÖH1:Süreç iyileştirme fırsatlarının tespit edilmesi

- ÖU1.1:Kurumsal süreç ihtiyaçlarının tespit edilmesi
- ÖU1.2:Kurumsal süreçlerin değerlendirilmesi
- ÖU1.3:Süreç iyileştirme fırsatlarının tespit edilmesi

ÖH2:Süreç iyileştirme aktivitelerinin planlanması ve gerçekleştirilmesi

- ÖU2.1:Süreç aksiyon planlarının oluşturulması
- ÖU2.2:Süreç aksiyon planlarının gerçekleştirilmesi
- ÖU2.3:Kurumsal süreç değerlerinin oluşturulması
- ÖU2.4:Süreçler ile ilgili tecrübelerin kurumsal süreç değerlerine katılmasının sağlanması

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaşması

OS3.7 Kurumsal Süreç Tanımlama

Amacı, kurumsal süreç varlıklarının kolay kullanıma uygun olarak oluşturulması ve güncel tutulmasıdır.

ÖH1: Kurumsal süreç varlıklarının oluşması

- ÖU1.1:Standart süreçlerin oluşması
- ÖU1.2:Hayat döngüsü model tanımlarının oluşturulması
- ÖU1.3:Uyarlama kriter ve rehberinin oluşturulması
- ÖU1.4:Kurumsal ölçüm havuzunun oluşturulması
- ÖU1.5:Kurumsal süreç varlık kütüphanesinin oluşturulması

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaşması

OS3.8 Kurumsal Eğitim

Çalışanların rollerini etkin ve verimli bir şekilde gerçekleştirebilmeleri için sahip olmaları gereken yetkinliklerin ve bilginin çalışanlar üzerinde oluşturulmasını amaçlar.[58]

Temel faaliyetler

Kurumun eğitim ihtiyaçlarının tespit edilmesi

İhtiyaçların giderilmesi için gerekli eğitimin sağlanması

Eğitim yeteneklerinin oluşturulması ve güncel tutulması

Eğitim kayıtlarının oluşturulması

Eğitim etkinliğinin ölçülmesi

ÖH1:Kurumsal eğitim yeteneğinin oluşturulması

- ÖU1.1:Stratejik eğitim ihtiyaçlarının oluşturulması

- ÖU1.2:Hangi eğitim ihtiyaçlarının kurumun sorumluluğunda olduğunun tespit edilmesi
- ÖU1.3:Kurumsal eğitim için taktik planın oluşturulması
- ÖU1.4:Eğitim yeteneğinin oluşturulması

ÖH2:Gerekli eğitimin sağlanması

- ÖU2.1:Eğitimin sağlanması
- ÖU2.2:Eğitim kayıtlarının oluşturulması
- ÖU2.3:Eğitimin yeterli olup olmadığının değerlendirilmesi

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaşması

OS3.9 Bütünleşik Proje Yönetimi

Projenin ve ilgililerin kurumun standart süreçlerinden özelleştirilmiş tanımlı ve bütünleşik bir süreç ile yönetilmesidir.[59]

Projenin ortak vizyonunun ve bütünleşik takım yapısının oluşturulması

Kritik tedarikçilerin proje aktiviteleri ile takip edilmesi

Ayrıca gereksinim oluşturma, tasarım, doğrulama, konfigürasyon yönetimi, dokümantasyon, pazarlama, eğitim gibi konuları da ele alır.

ÖH1:Projenin tanımlı süreçlerinin kullanılması

- ÖU1.1:Projenin tanımı süreçlerinin oluşturulması
- ÖU1.2: Proje planlama aktivitelerinde kurumsal süreç varlıklarının kullanılması
- ÖU1.3:Planların bütünleştirilmesi
- ÖU1.4:Projenin bütünleşik planlar ile yönetilmesi
- ÖU1.5:Kurumsal süreç varlıklara katkıda bulunulması

ÖH2:İlgili paydaşlar ile işbirliği ve koordinasyon içinde bulunulması.

- ÖU2.1:Paydaşların katılımının yönetilmesi
- ÖU2.2:Bağımlılıkların yönetilmesi
- ÖU2.3:Koordinasyon sorunlarının çözülmesi

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaşması

OS 3.10 Bütünleşik Takım Oluşturma

İş ürünlerinin bütünleşik bir yapıda olmasını sağlamayı amaçlar.

Takım üyeleri

Takıma gerekli olan uzmanlık ve yetkinliklerin geliştirilmesi

Takımda geldikleri noktaların tespiti

Takım ve takım dışı paydaşlar ile iş birliği içinde bulunurlar

Takım işleri ve hedefleri için ortak bir anlayış geliştirirler.

Takımın belirlemiş çalışma kurallarına ve prensiplerine uyarlar.

ÖH1:Takım üyelerinin belirlenmesi

- ÖU1.1:Takım'ın görevlerinin belirlenmesi
- ÖU1.2:Gerekli bilgi ve yetkinliklerin belirlenmesi
- ÖU1.3:Uygun takım üyelerinin atanması

ÖH2:Takım çalışmalarının yönetilmesi

- ÖU2.1:Ortak bir görüş(vizyon tespiti) belirlenmesi
- ÖU2.2:Takım beratının/ tüzüğünün oluşturulması
- ÖU2.3:Görev ve sorumlulukların tanımlanması
- ÖU2.4:Çalışma prosedürlerinin belirlenmesi
- ÖU2.5:Takımlar arası iş birliğinin sağlanması

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaşması

OS3.11 Bütünleştirme için Organizasyonel Ortam

Bütünleşik ürün ve süreç geliştirme için gerekli altyapıyı oluşturmak ve çalışanların bütünleştirmeye yönelik olarak yönetilmesidir.[59]

<u>Temel Bileşenler</u>

Kurumsal vizyon

Liderlik mekanizması

Bütünleştirme için teşvik sistemi

Kurum ve takım görevlerinin dengelenmesi için mekanizmalar

ÖH1:Bütünleşik üretim altyapısının oluşturulması

- ÖU1.1:Kurumsal ortak görüşün oluşturulması
- ÖU1.2:Bütünleşik çalışma ortamının oluşturulması
- ÖU1.3:Bütünleşik üretim için gerekli yetkinliklerin tespiti

ÖH2:Bütünleşik üretim için insanların yönetilmesi

- ÖU2.1:Liderlik mekanizmalarının oluşturulması
- ÖU2.2:Bütünleşik üretim için teşviklerin oluşturulması
- ÖU2.3:Takım ve kurum sorumluluklarının dengelenebilmesi için mekanizmaların oluşturulması

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaşması

OS3.12 Bütünleşik Tedarikçi Yönetimi

Proje isterlerini yerine getirmek için ürün alımında kullanabilecek tedarikçilerin proaktif olarak tespit edilmesi ve seçilen tedarikçilerin proje işbirliği içerisinde yönetilmesi.[60]

Temel Faaliyetler

Potansiyel tedarikçinin tespit edilmesi, amaliz ve seçimi

Ürün alımında kullanılacak tedarikçilerin değerlendirilmesi

Seçilen tedarikçi süreçlerinin değerlendirilmesi

Seçilen tedarikçi ürünlerinin değerlendirilmesi

Gerektiğinde tedarikçi sözleşmelerinin ve ilişkilerinin gözden geçirilmesi

ÖH1:Olası tedarikçilerin analiz edilmesi ve seçilmesi

- ÖU1.1: Olası tedarikçilerin analiz edilmesi
- ÖU1.2:Tedarikçilerin değerlendirilmesi ve seçilmesi

ÖH2:Tedarikçiler ile koordinasyon sağlanması

- ÖU2.1:Tedarikçilerin seçilen süreçlerin izlenmesi
- ÖU2.2:Tedarikçi seçilmiş iş ürünlerinin izlenmesi
- ÖU2.3:Tedarikçi sözleşmelerinin ve ilişkilerinin gözden geçirilmesi

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaşması

OS3.13:Risk Yönetimi

Olası problemler oluşmadan tespit ederek, bu problemlerin proje hedeflerine ulaşılmasına engel olmasını engellemek için, önlemlerin planlanması ve hayata geçirilmesidir.[61]

Temel Faaliyetler

Bir risk yönetim stratejisinin belirlenmesi

Risklerin tespit edilmesi ve analizi

Tespit edilen riskler için önlemler alınması

ÖH1:Risk yönetimi için hazırlıkların yapılması

- ÖU1.1:Risk kaynak ve kategorilerinin belirlenmesi
- ÖU1.2: Risk parametrelerinin belirlenmesi
- ÖU1.3: Risk yönetim stratejisinin belirlenmesi

ÖH2:Risklerin tespit edilmesi ve analizi

- ÖU2.1:Risklerin tespit edilmesi
- ÖU2.2: Risklerin değerlendirilmesi, kategorize edilmesi ve ilklendirilmesi

ÖH3: Risk olasılıklarının azaltılması

- ÖU3.1:Risk olasılıklarının azaltma planlarının geliştirilmesi
- ÖU3.2: Risk olasılık azaltma planlarının hayata geçirilmesi

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaşması

OS 3.14 Karar Analiz ve Çözüm Üretme

Olası kararların, belirlenmiş kriterlere göre değerlendirilmesi ve resmi bir değerlendirme süreci kullanılarak, analiz edilmesidir.[62]

Temel Faaliyetler

Alternatiflerin değerlendirilmesi için kriterlerin belirlenmesi

Alternatif çözümlerin belirlenmesi

Alternatiflerin değerlendirilmesi için yöntemlerin seçilmesi

Alternatiflerin belirlenmiş kriterlere göre değerlendirilmesi

Alternatifler arasından kriterlere uygun olarak çözümlerin geliştirilmesi

ÖH1: Alternatiflerin değerlendirilmesi

- ÖU1.1:Karar analiz için rehber olarak kullanılabilecek prensiplerin geliştirilmesi
- ÖU1.2:Değerlendirme kriterlerinin belirlenmesi
- ÖU1.3: Alternatiflerin çözüm önerilerinin belirlenmesi.
- ÖU1.4:Değerlendirme yöntemlerinin seçilmesi
- ÖU1.5: Alternatiflerin değerlendirilmesi
- ÖU1.6:Çözümün seçilmesi

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaşması

4.11 CMMI 4 Seviye Süreçleri

OS4.1 Kurumsal Süreç Performansı

Süreç performans ve kalite hedeflerine desteklemek için, kurumun standart süreçleri hakkında veriler toplamak ve toplanan verileri güncel tutmak. Projenin sayısal yöntemi için ihtiyaç duyulacak olan süreç performans verilerini, temel sürümleri ve modellerini sağlamaktır.[63]

ÖH1:Performans temel sürümleri ve modellerinin oluşturulması

- ÖU1.1: Süreçlerin seçilmesi
- ÖU1.2:Sürçe performans ölçülerinin belirlenmesi
- ÖU1.3:Kalite ve süreç performans ölçülerinin belirlenmesi
- ÖU1.4:Süreç performans temel sürümlerinin oluşturulması
- ÖU1.5:Süreç performans modellerinin oluşturulması

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaşması

OS4.2 Sayısal Proje Yönetimi

Belirlenmiş kalite ve süreç performans hedeflerine ulaşması için, projenin tanımlanmış süreçlerinin sayısal olarak yönetilmesidir. Tedarikçi kullanılması hainde, bütünleşik tedarikçi yönetimi ve tedarik sözleşme yönetimi süreç alanları ile koordinasyon sağlamalıdır.

<u>Temel Faaliyetler</u>

Projenin kalite ve süreç performans hedefleri belirlenmesi

Kurumsal veri tabanlarında verileri tutulan alt-süreçlerin tespit edilmesi

Proje sayısal olarak yönetilecek alt-süreçlerin belirlenmesi

Belirlenen alt-süreçlerin sayısal yönteminde kullanılacak ölçüm ve analiz yöntemlerinin belirlenmesi

Belirlenen alt-süreçlerin projenin belirlenmiş kalite ve süreç performans hedefleri doğrultusunda sayısal olarak yönetilmesi

ÖH1: Projelerin sayısal olarak yönetilmesi

- ÖU1.1:Projenin hedeflerinin tespit edilmesi

- ÖU1.2:Tanımlı süreçlerin bir araya getirilmesi
- ÖU1.3:İstatiksel olarak yönetilecek alt süreçlerin

ÖH2:Alt süreçlerin performansının istatiksel olarak yönetilmesi

- ÖU2.1:Ölçüt ve analitik tekniklerin seçilmesi
- ÖU2.2:Değişkenliği anlamak için istatiksel yöntemlerin kullanılması
- ÖU2.3:Seçilen alt süreçlerin performansının gözetlenmesi
- ÖU2.4:İstatiksel yönetim verilerin kayıt altına alınması

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaşması

4.12 CMMI 5 Seviye Süreçleri

OS5.1 Kurumsal Yaratıcılık ve Yaygınlaştırma

Kurumsal süreçler ve teknolojide ölçülebilir iyileştirme sağlamak için yaratıcı iyileştirme önerilerinin seçilmesi ve seçilen önerilerin kurum geneline yaygınlaştırılmasıdır. İyileştirmeler, şirketin iş hedeflerine uygun olarak oluşturulmuş olan kurumsal kalite ve süreç performans hedeflerine ulaşmalarında destek olmaktadır.

Örnekler:[64]

Ürün kalitesinin iyileşmesi

Verimliliğin artması

Üretim sürelerinin azalması

Müşteri ve son kullanıcı memnuniyetlerinin arttırılması

ÖH1:İyileştirme fırsatların seçilmesi

- ÖU1.1: İyileştirme önerilerinin toplanması ve analiz edilmesi
- ÖU1.2:Yaratıcı fikirlerin tespit edilmesi ve analiz
- ÖU1.3:Pilot iyileştirme çalışmaları

ÖH2:İyileştirme çabalarının yaygınlaştırılması

- ÖU2.1: Yaygınlaştırılmanın planlanması
- ÖU2.2: Yaygınlaştırılmanın yönetilmesi
- ÖU2.3:İyileştirmenin etkilerinin ölçülmesi

GH2:Yönetilir bir sürecin kurumsallaştırılması

GH3:Tanımlı bir sürecin kurumsallaşması

OS5.2 Sebep Analiz ve Çözüm

Hataların ve problemlerin sebeplerini tespit ederek gelecekte tekrar oluşmalarını engellemek için gerekli aksiyonların alınmasını sağlamaktır.

ÖH1:Hataların sebeplerin saptanması [

- ÖU1.1:Analiz edilecek hata verilerinin seçilmesi
- ÖU1.2:Sebeplerin analiz edilmesi

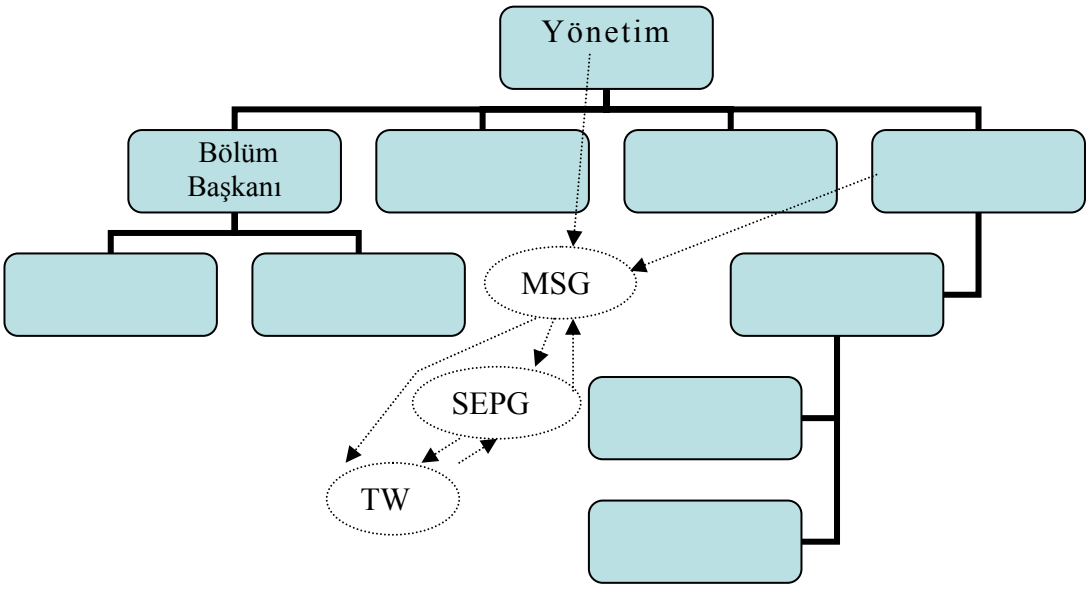
ÖH2:Hataların sebeplerinin bulunması

- ÖU2.1:Aksiyon önerilerinin hayata geçirilmesi
- ÖU2.2:Değişikliklerin etkilerinin değerlendirilmesi
- ÖU2.3:Verilerin kayıt edilmesi

GH2:Yönetilir bir sürecin kurumsallaştırılması

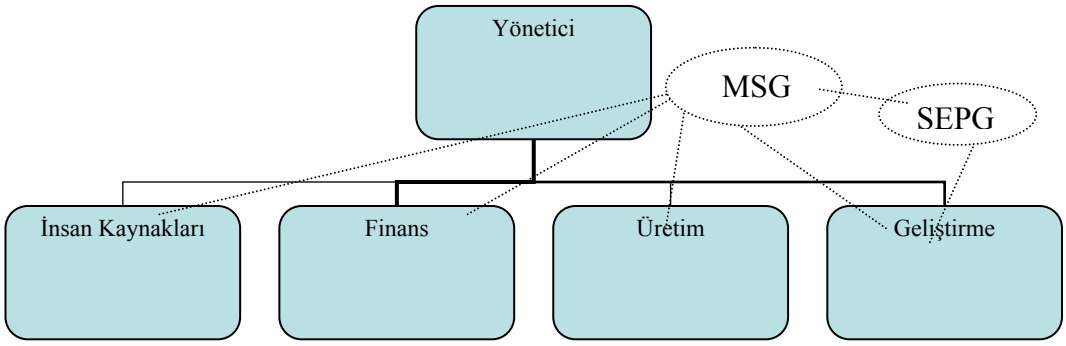
GH3:Tanımlı bir sürecin kurumsallaşması[65]

4.13 SPI Yapısı



Şekil 4.12 SPI Yapısı

MSG(Management Steering Group):Yönetici
SEPG(System Engineer Process Group) : Şirket içerisinden olması
TWG(Team Working Group)[66]



Şekil 4.13 SPI yapısı ve diğer bölümler ile ilişkisi

4.14 CMMI-Süre

Seviye artırımı için geçen süreler aşağıdaki tablo da verilmiştir.

Tablo 4.7 CMMI Seviye Geçiş Süreleri

Seviyeler	**Ortalama**	**En Az**	**En çok**
1.Seviyeden ikinci seviyeye geçiş	20 ay	18	46
2.Seviyeden üçüncü seviyeye geçiş	19 ay	17	30
3.Seviyeden dördüncü seviyeye geçiş	25 ay	18	44
4.Seviyeden beşinci seviyeye geçiş	13 ay	10	22

4.15 CMMI sahibi olan şirketlerin en çok karşılaştıkları sorunları

CMMI'ın ne olduğunu anlamak için CMMI projelerinde çok sık karşılaşılan sorunlara da bakmak gerekir.

CMMI çalışmalarında en çok görülen yedi risk şunlardır:

1. İş Hedefleri – CMMI ilişkisi
2. Aşırı Araç Odaklılık
3. Değişimin kalıcı olmaması
4. Üst Yönetimin Süreksizliği
5. İnsanların Mantığına aşırı güven
6. Uzun ve kısa vadeli hedeflerin dengelenememesi
7. Sorumlu kişilerin yanlış seçimi [69]

4.15.1 İş Hedefleri – CMMI ilişkisi

CMMI sertifikasını alan şirketlerin üst yönetimleri üç önemli soruyu net bir biçimde yanıtlamaları gerekmektedir.Eğer bu üç soruyu yeteri kadar net bir biçimde yanıtlamalıdırlar.

Ne: CMMI projelerinin yapılarının detaylı bir şekilde tanımlanmış olmasına özen gösterilmelidir. Bölümlerin hangi projeleri hangi teknolojik alt yapıyla ne kadar bir kaynak kullandıklarının iyi belirlenmiş olmasına özen gösterilmelidir.

Neden: Yapılan işlerin işi ilgilendiren bütün kesimlere; müşteriler, hissedarlar, yöneticiler ve çalışanlar için neden önemli olduğunun iyi bir şekilde anlatılması gerekmektedir.

Nasıl: Yapılan işin yapılma kriterlerin ortaya konularak ölçütlerinin belirlenmesi gerekmektedir.

CMMI'da başarı sertifika almak değil alınan bu sertifikaya göre işlerin daha hızlı daha az maliyetle daha az hata da oluşturulmasını sağlamaktır.

4.15.2 Her şeyi araçlardan beklemek

CMMI projeleri insanların davranışlarının değiştirilmesi ile ilgilidir. Sadece bir araç kullanılıp sonuca gidilmesini beklememek gerekir. Şirketler liderlik vasıflarını kararlılıkla sergilemelidir.

4.15.3 Değişimi işin bir parçası yapmak

Yazılım işlerinin uzun soluklu işler olduğunu bilerek şirketinizi hazırlayın sadece tek bir proje bazında hazırlanmayın günlük sorunlar yüzünden uzun vadeli planlarınızı ertelemeyin.

4.15.4 Üst Yönetimin süreksizliği

Üst yönetim projeler içerisinde bulunarak; sürekli katılım, kararlılık, kontrol ve takip gibi yöntemler ile işlerin gidişatını değerlendirmelidir. Üst yönetimin hedefleri belli aralıklar ile yenilenmelidir.

4.15.5 İnsanların mantığına aşırı güven

İnsan davranışlarında, istenmeyen bazı nedenlerden dolayı sorunlar olabilir. Bu sorunlara karşı bir insan kaynakları planı hazırlanmalıdır. Davranışlarının değişmesi gereken kişiler var ise bu kişiler tespit edilmelidir. Değişimin yararları anlatılmalıdır. Değişimi engelleyen olaylar ya da kişiler belirlenmelidir.

4.15.6 Uzun ve kısa vadeli hedeflerin dengelenememesi

CMMI çalışmalarının somut sonuçların görülmesi için 2 seneye ihtiyaç vardır. Bu süreyi ana hedefle uyuşan ara hedefler belirlenmelidir. Çalışmalarında 6 aylık sürelerle belli hedefler belirlenerek gelişimler izlenmelidir.

4.15.7 Sorumlu kişilerin belirleme hataları

Her hangi bir aksaklık durumunda işin sorumluluğunu üstlenecek kişiler en baştan belirlenmelidir.

4.16 CMMI Süreç Tasarımı

Sürecin mükemmel olamayacağı bilinci ile hareket edilir. Performans hedefleri belirlenir. Yapılacak çalışmanın genele ulaştırılmadan kurulacak bir test grubu ile pilot bölge üzerinde gözden geçirilir. Dokümantasyon ve eğitim materyalleri hazırlanır. Pilot ve test grubu oluşturulur. Mevcut sistem özellikleri belirlenir. Çözüm teknik ekip ile birlikte hazırlanmaya başlanır. Ekip içinde roller saptanır. Çözüm içinde amaç belirlenir. Bakış acısını geliştirmek için proje içerisinde kullanılan bir metafor oluşturulur. Müşteriler ile bir araya gelinerek oluşturulacak ürün ile ilgili bazı temel fikirler üzerinde durulur. Bu görüşme sonucunda oluşturulacak projeyi kimin kullanacağı, müşterinin ne gibi beklentileri olduğu hesaplanır.

Basitlik süreç gelişimi için önemli bir faktördür. Süreçleri basitleştirmek için yazılımı geliştiren ekip üç konuya dikkat etmelidir. Süreçleri geliştirmek için çok mu mükemmeliyetçi davranıyoruz? Süreç içerisindeki bu adım olmaz ise ne kaybederiz? Süreç tanımlarını yaparken iş akışlarına dikkat ediyor muyuz?gibi sorulara dikkat edilmelidir.

Süreç tasarımlarında dikkat edilecek diğer bir husuz ise uygulanabilirliktir.

Süreçlerin adımlarının projenin gelişimi açısından uyun olup olmadığı ile ilgilidir. Uygulanabilirlik için önemli olan husuzlar ise; eğitim ihtiyacı, araç desteği(veriler hangi ortamda duracak),zaman, bütçe, kaynaktır.

4.16.1 CMMI-Ölçeklenebilirlik

Tablo 4.8 CMMI Ölçeklenebilirlik

Şirket Hedefi	**Proje Konuları**	**Süreç Konuları**	**Ürün ve süreç özellikleri**
Fonksiyonların arttırılması	Ürün gelişimi Ürün kararlılığı	Ürünün gereksinimlere uygunluğu	Gereksinim sayısı Ürün boyutu/büyüklüğü Ürün karmaşıklığı Değişim oranı Hata oranı
Maliyetlerin azaltılması	Bütce harcama oranları.	Etkinlik verimlilik	Ürün boyutu/büyüklüğü Ürün karmaşıklığı Emek Değişiklik sayısı
Ürün teslim zamanının kısaltılması	Zaman planı ilerleme	Üretim hızı Tepki verme süresi	Ürün karmaşıklığına göre düzeltmek için harcanan zaman
Ürün kalitesinin arttırılması	Ürün performansı Ürün doğrulama Ürün güvenliği	Tahmin edilebilirlik Sorun saptama Sorun sebep analizi	Üretilen hata sayısı Hata bulma faaliyetleri etkinliği Ortalama hatasız çalışma süresi

[71]

4.17.2 CMMI –Değerlendirme

SCAMPI (Standart CMMI Appraisal Method for Process Improvement)

- Değerlendirmeyi yapacak takımın belirlenmesi ve eğitimlerinin verilmesi.
- Olgunluk anketleri (CMM den örneklemeler)
- Cevapların analizi
- Saha ziyaretleri (mülakatlar ve dokümanların incelenmesi)
- Bulgular (CMM pratikleri uyarınca elde edilen bulgular)
- Süreç profilleri
- Tavsiyeler (Bulgular ve süreç profili baz alınması)
- Hareket planı saptanır.(Tavsiyeleri hayata geçirme kısmı)

Değerlendirme Çeşitleri

Tablo 4.9 Değerlendirme Çeşitleri

Özellik	A Sınıf	B Sınıfı	C Sınıfı
Genel Ölçüm sonuçları	Üretir	Üretilmez	Üretilmez
Ekip büyüklüğü	Büyük	Orta	Küçük
Değerlendirme Ekip lideri	Uzman	Uzman ya da eğitilmiş ve tecrübeli	Eğitilmiş veya tecrübeli

Değerlendiriciler

Tablo 4.10 CMMI Değerlendirciler

Sponsor	Organizasyon yöneticisi
Koordinatör	Değerlendirme sürecini kontrol eden organizasyon sorumlusu
Baş Denetleyici	Uzman değerlendirici sertifikasına sahip değerlendirme ekip lideri
İç Denetleyici	Değerlendirme ekibi ile beraber değerlendirme faaliyetlerine katılacak eğitim organizasyon uzmanları
Dış Denetleyiciler	Değerlendirme ekibini oluşturan ve organizasyon çalışanı olmayan değerlendiriciler

Değerlendiriciler ekibinin büyüklüğü en az 4 en çok 9 kişiden oluşmalıdır. Ekip ortalama olarak en az 6 yıl mühendislik deneyimine sahip olmalıdır. Ekip toplam olarak en az 10 yıl yönetim deneyimi olmalıdır.

Değerlendirme Modeli:

Doğrudan: Bu değerlendirme sonucunda gözle görülen değerlendirme bilgilerine ulaşılır.

Dolaylı: Pratiklerin yerine getirildiğine dahil bilgiler olan ama ana pratiğin gerçekleşmesine bağlı olmayan göstergeler.

Onaylar: Pratiklerin gerçekleştiği bilgisi sözlü olarak alır.

Örnek: Proje Planlama

ÖU1.1: Projenin kapsamına yönelik ön kestirimler için bir üst düzey iş detaylandırma yapısı geliştir.

Doğrudan: Üst düzey iş detaylandırma yapısı ve değişiklik tarihçesi, görev tanımları ve iş ürünü tanımları

Dolaylı: İş detaylandırma yapısı oluşturmak için yapılan toplantı tutanakları.ÖU ile ilgili proje ön kestirimleri

Onaylama: İş detaylandırma yapısı nasıl kullanılıyor ve ön kestirimler nasıl oluşturuluyor?

Temel Değerlendirme modeli içerisinde

Tablo 4.11 CMMI Ölçeklenebilirlik

Yeterlilik seviyesi ve/veya olgunluk seviyesi ölçümü	Bütün ekip
Hedef gerçeklenme ölçümü	Bütün ekip
Pratik gerçeklenme belirlemeleri (organizasyonel birim seviyesi)	Bütün ekip
Pratik gerçeklenme belirlemeleri (süreç pratiği uygulama seviyesinde)	İlgili birimler 4-9 kişilik asıl ekip tarafından değerlendirilir.

Değerlendirme sonuçlarının üretilmesi

Süreç uygulaması,pratik gerçeklenme verileri,pratik gerçeklenme değerlendirilmesi

Değerlendirme sonucunda bir değerlendirme raporu çıkartılır bu raporda: Değerlendirme Sonuçları, analizler(artı ve eksi olan yönlerin belirlenmesi,

geliştirme önerileri) süreç alan profilleri belirlenir(süreç alanları yeterlilikleri, süreç uygulamaları yeterlilikleri, organizasyon el seviyede süreç yeterlilikleri)ve en son olarak ta organizasyonel anlamda süreç in "olgunluk düzeyi" bilgisi verilir.

4.18 CMMI - Yüksek Olgunluk

Yazılımların süreçlerini iyileştirmeyi amaçlayan Carnegie Mellon Üniversitesi'nin Yazılım Mühendisliği Enstitüsü tarafından oluşturulan CMMI(Capability Maturity Model Integration) organizasyonları yazılım süreçlerini yönünden 5 aşamalı bir şekilde tanımladığından bahsetmiştik. Firmalar kendi yazılım süreçlerine ayit notları arttırdıklarında o seviyelerin anahtar işlevlerini gerçekleştirirler. Firmaların amacı bütün bu anahtar süreçleri bitirip en iyi olarak tarif edilen 5 inci düzeye çıkmaktır. Eğer firmalar 4. veya 5. seviyeye çıkabilirlerse kendi iş yaptıkları alanlarda diğer rakiplerine de büyük bir çapta fark yaratmış olacaklardır. Bu yüksek seviyelerde not alan firmalar yani yüksek olgunluğa sahip firmalara bakıldığında bazı ortak noktalara sahip oldukları görünmüştür.

Bu yüksek olgunluk düzeyine gelen firmaların detaylı olarak özelliklerine bakıldığında bazı özelliklerinin ortak olduğu görülmüştür. Aşağıda yüksek olgunluğa sahip firmaların hangi teknikleri ve hangi yöntemleri kullanarak yüksek olgunluk düzeylerine sahip olduklarıyla ilgili bilgiler verilmiştir. Bu bilgilerde şirketlerin kendi sahip oldukları ve onları diğer firmalardan daha üstün kılan özellikler olarak ta değerlendirilebilir. [72]

Yönetim Uygulamaları Hemen Hepsi;

- Süreçleri geliştirme sırasında CMM ın çalışma alanından daha geniş olarak işe yaklaşıyorlar
- Bağımsız çalışabilen bir SEPG una sahipler
- Kaliteyi sürekli olarak yüksek seviyede tutmak için süreçleri oluştururlarken kaliteyi sağlayacak yapıları içerisine entegre edebiliyorlar
- Proje içerisindeki kişilere süreçlerin ve ürünlerin ölçülmesi için değişik yöntemler kullanılması sağlanıyor.
- Süreç ile ilgili dokümantasyonlara istendiği zaman erişiliyor

Çoğunluğu;

- Başlangıçta ISO 9001 sertifikasına sahip ve daha sonradan CMM modelini kullanmaya başlamış.
- Süreçler geliştirilme ve TQM konuları bir arada ele alınıyor
- Gömülü sistemler geliştiriliyor.
- Maliyet tahminlemesi için COCOMO gibi araçlardan faydalanılıyor.
- Yapılan işlerin maliyet hesapları detaylı olarak çıkarılıyor.
- Riskler ve felaket dutrumu gibi konular hakkında raporlar hazırlanıyor oluşabilecek risklerin neler olabileceği hesaplanmaya çalışılıyor.
- Proje için PERT diyagramları kullanılıyor.
- Mevcut süreçlerle ilgilenecek kişiler belirleniyor danışmanlar ve gözlemciler belirlenir.
- Projede kullanılacak veriler teftiş amaçlı olarak gözden geçirilir bu sayede yeni bilgi ihtiyaçları da belirlenir.
- Yapılacak yazılımların arayüzlerinin birer prototipi hazırlanır.
- Verilerin tutarlılığından ve işlenen verilerin doğruluğundan emin olunması için bir test grubu bulundurulur.
- CASE araçlarından faydalanır.

En az yarısı;

- Delphi metotlarını kullanılarak tahminle yapılıyor.
- Süreçlere bizzat kalıtmış ve destek vermiş kişilerle birlikte süreçler gözden geçirilir.

Bu ve buna benzer bazı uygulamaları gerçekleştiren şirketler sahip oldukları kaliteyi korudukları gibi başka teknik ve modelleri de kullanarak kaliteyi daha üst seviyelere taşıyabilirler. Sonuç olarak değerlendirdiğimizde, bu yüksek olgunluğa sahip firmalar ne yaptıklarını ve nasıl yapacaklarını son derece iyi bilen şirketlerdir. Sorunların üstesinden diğer şirketlere göre daha fazla deneyimlere sahip olduklarından ve benzer sorunlarla karşılaştıklarından bu problemle iyi baş edebilirler. Özet olarak meydana getirile bilecek hataların minimum düzeye çekilmeye çalışıldığı sorumlulukları kendi içerisinde alıp işi gözünde büyütmeden taşın altına elini rahatça sokan yetkilerin ve yapılacak işlerin önceden son derece detaylı olara ka tanımlandığı hedeflere nasıl ulaşılacağını bilen şirketlerdir diye biliriz.

5.Extreme Programming

5.1 XP nedir?

XP yazılım geliştirmesine yeni bir bakıştır. Yaklaşık yedi yıllık bir geçmişe sahiptir. XP içeriği gereği dinamik ve riskli projelerde uygulanabilen bir yapısı vardır. XP in en önemli avantajı ise müşterilerle kurduğu çok yakın ilişkidir. Bu sayede müşterilerin ihtiyaçları algılanıp, gerektirdiği şekilde çözümlenmeye çalışılır.

XP 1990'ların sonlarında yazılım dünyasında çeşitli eleştirilerin yüksek sesle dile getirilmesiyle ortaya çıkmıştır. [73]

Bu eleştiriler;

1. Sürüm yani çalışan program çok geç ortaya çıkıyor.
2. Sürüm çok geç çıktığı için hatalar çok geç anlaşılıyor.
3. Verimli değil.
4. Esnek değil, yeni gelen isterler doğrultusunda kendi yapısını değiştiremediği için yeni isterleri karşılayamıyor.
5. Değişiklik geç ve zor yapılıyor.

Bu ve buna benzer eleştiriler doğrultusunda 1990'ların sonlarında değişik modellerde çıktı.

Bu çıkan alternatif modeller, yazılım geliştirme işine; analiz, tasarım, sınama, entegrasyon ve bakım aşamalarına daha farklı bir bakış açısı ile yaklaşmaktaydılar. Bu alternatiflerden biriside XP idi.

XP (extreme programming) ile kez 1999 yılında Kent Beck Tarafından duyuruldu. XP (extreme programming) ya da diğer adıyla çevik metotlar çağlayan modeline karşı oluşan eleştirileri temel alınarak geliştirilmiştir.[74]

Bir yazılım geliştirme disiplini olarak ortaya çıkan Extreme Programming in amacı; kolay, iletişimin daha çok kullanıldığı, geri dönüşlerin daha fazla olmasına imkân sağlayan bir yazılım geliştirme yöntemi olmaktı. Kendini cesaretle yazılım geliştirme dünyasında tanıttı.

Günümüzde pek çok şirketi tarafından kullanılan XP (Bayerische Landesbank, Credit Swiss Life, DaimlerChrysler, First Union National Bank, Ford Motor Company and UBS.) kullanılmaktadır. XP projelerinin daha hızlı bir şekilde gerçekleştirilmesini sağladı. XP Yazılım geliştirme sırasında bütün takımın aynı amaç doğrultusunda bir arada olma prensibi gibi kendisine has yöntemleri vardır. Böylece diğer yazılım geliştirme yöntemlerine göre çok farklı yazılım geliştirme özellikler taşımaktadır.

5.1.1 XP Tarihi

XP nin tarihine bakmadan önce proje gelişiminin tarihine bakmamız gerekir.
Proje yönetiminin tarihi[75]

Tablo 5.1 Proje Yönetimi Tarihi

Dönemler	Tema	İçerik
1958 öncesi dönem	Craft(el sanatı) sistemi insan ilişkileri dönemi	Proje Yönetimi Aktüel projeler
1958–1979	Yönetim bilimi uygulamaları	
1980–1994	Üretim merkezli insan kaynakları	
1995 ve sonrası	Yeni çalışma çevresi oluşturulması	

(Proje yönetiminin dört farklı boyutu)
XP tarihsel gelişimi ?
1970 Bireysel XP uyuglamaları
1979 Alexander (Karar verme)
1986 Hızlı gelişme (Nonaka ,Takeuchi and cunningham)
1988 Evrimsel Gelişme (Gilb)
1988 Spiral Model (Bohem 1988)
Proje Özellikleri Planlama (Jacobsen 1994)
Metafor (Lakoff, Johnson 1998)
Programcılar için fiziksel ortam (De morca, Coplien 1999) [76]

5.1.2 XP'nin Bazı Avantajları

- Yazılımın erken safhalarında somut gelişmeler sağlayabilir. Oluşan hataları erken safhalarda farkına varılır. Bu sayede küçük yaşam döngüleri oluşturarak geri dönüşler yapar ve hatalar telafi edilir.
- Artırımsal planlama yaklaşımı sayesinde hızlı bir şekilde genel planın oluşturulmasını sağlar. Artırımsal yaklaşımla birlikte projenin gelişip, hayata geçmesi süreci tahmin edilmeye çalışılır.
- Esnek iş yürütme ve fonksiyonellik sayesinde, işletmelerin değişen ihtiyaçlarına cevap verilir.
- Program geliştirilme süresince monitörlerden programcılar ve müşteriler ortak bir şekilde test yaparak, programı güvenliğini ve istenilenlere uygun olup olmadığını anlamaya çalışırlar. Bu sayede sistem gelişimi sağlanır ve erken safhada ileride olabilecek bir yazılımsal kilitlenmeyi ortadan kaldırır.

XP 2001'de yazılım mühendisliğine farklı bir bakış açısı getiren bir manifesto yayınladı.

Bu manifestoya göre;

1-) Gerçekleştirilecek yazılımı kullanacak kişiler müşterilerdir. Müşterilerin yazılım geliştirilirken ki önerileri çok önemlidir. Bu yüzden müşteri ile yazılımı gerçekleştirecek kurumlar arası ilişkiler son derece sıcak olmalıdır.

2-) Çalışan bir yazılımın kapsamlı belgelerden daha çok müşterilerin ihtiyaçlarını karşılaması önemlidir. Yüz yüze görüşme alternatif dokümantasyondan daha önemlidir.

3-) Yazılımı yapan kişilerle müşteriler arasındaki ilişkiler daha sık olmalıdır.

4-) Yapılacak iş planı değişkendir. Plan izlemektense değişikliklere bakarak yol belirlenir.

5-) Müşteriye erken ürün vererek müşteri doyurulmalıdır.

6-) Değişen isterlere anında cevap vermek çok önemlidir.

7-) İşin büyüklüğüne göre 2 ayla 2 yıl arası sürüm ortaya çıkar.

8-) İş yapmak için motive olmuş insanlara gerek vardır. İş, motive olmuş insanlara verilmelidir.

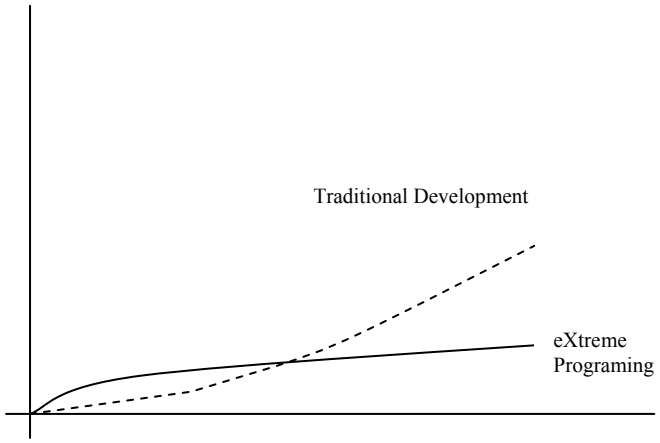
9-) Gelişmenin en iyi ölçüsü, gelişen yazılımdır.

10-) Basitlik, en temel ilkedir.

11-) Yazılımı gerçekleştirecek organizasyonlarda en iyi organizasyon 10 -15 kişi arasıdır.

12-) Belli aralıklarla durum değerlendirmesi yapılmalıdır.[77]

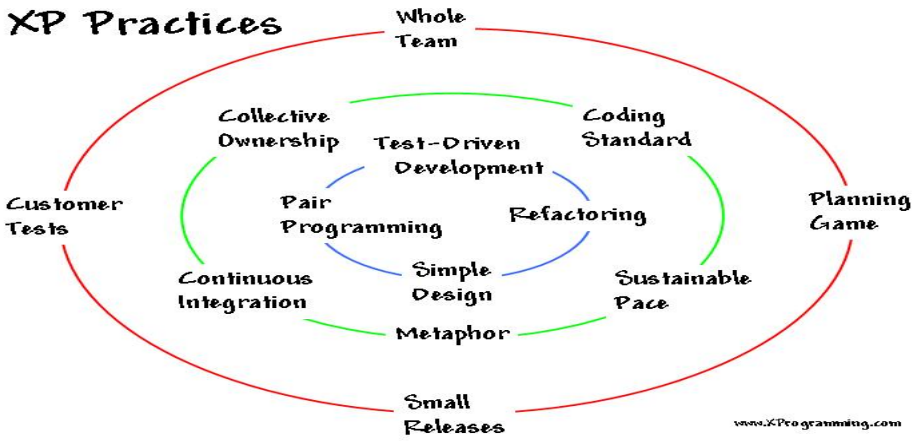
Tablo 5.2 Değişim Maliyeti ile XP in İlişkisi

<table>
<tr><td colspan="2">Değişim Maliyeti</td></tr>
<tr><td> 
</td><td>Değişimin maliyeti konusuna baktığımızda geleneksel yaklaşımda değişim maliyeti artarken Extreme Programming’te o oranda bir artış olmadığını görüyoruz.</td></tr>
<tr><td>Analiz
Tasarım
Kodlama
Bakım</td><td></td></tr>
</table>

5.2 XP in çalışma mantığı

XP in çalışma mantığı, bazı basit pratiklere ve uygulamalara bağlıdır. Başlangıçta bu pratikler çok basit ve gereksiz gelse de bir süre geçtikten sonra alınan başarılar ile kendi gücünü gösterir.

Müşteriler ile birlikte yazılım geliştiricilerin aktif olarak proje içerisinde bulunması sayesinde, projenin uzmanlaşmış bir seviyeye ulaşması sağlanır. XP yaşam döngüsü içerisinde proje içerisindeki değişikliklerin yapılmasın konusunda yazılımcıya güven verir. Bu metodolojide takım çalışması vurgulanır. Müşteriler ile birlikte alınan kararlar doğrultusunda işler yürütülür. Geliştirmede en uygun yol belirlenir. Bir XP yazılım projesini 4 temel yoldan geliştirir. İletişim, basitlik, geri dönüşüm ve yazılımda cesarettir.



Şekil 5.1 XP Pratikleri

Şekil 5.1 de gösterilen XP pratiklerinde dış kısımda müşteriler ve teknik ekibin teknikerin bir arada eş güdümlü çalışması sağlanır.

Orta kısımda iş sorunlarına ilişkindir. İş sorunlarına odaklanılmasını sağlar. Teknik sorunların ortadan kaldırılması ile ilgilidir.(Ör: Ne kadar zamanda bir sürüm alınacak)

İç kısımda yazılımın kalitesinin düşmemesini sağlamak amaçlı olarak oluşturulmuştur.Yazılım üretme aşamasında kalitenin düşmemesini sağlar. Üretim kısmı ile ilgili.

Extreme Programming de, projeye küçük ya da büyük bir katkı sağlayan her kez proje içersinde ayrılmaz bir bütün olarak görülür. Müşteri de bu takım çalışmasının bir parçasıdır. Ürün geliştirilme sırasında müşteriler de aynı birer çalışan gibi görevler alarak üretilecek yazılım grubu içerisinde bulunurlar.

Yaşam döngüsünde XP in ana ilkesi, işler bir anda yapılmayacak. İşleri parça yapalım. Arada sırada yaptığımız işlere bakalım değişiklik gerekiyorsa değişiklik gereken yerleri yıkıp tekrardan yapalım Projenin ilerleme hızları daha çabuk olmaya başlayacaktır.[78]

5.3 XP in 4 Temel Noktası

5.3.1 İletişim:

XP in ilk temel taşı iletişimdir. Projelere baktığımızda ortaya çıkan önemli problemlerin insanların birbirleriyle tam olarak anlaşamaması nedeniyle olduğunu görürüz. Bazen programcılar sormaları gereken doğru soruyu soramazlar. Bazen de projenin yapısıyla ilgili önemli bir gelişmeyi söylemezler. Bu yüzden projelerde çeşitli tıkanma noktaları olabilir. Bazen de proje yöneticileri programcılara belirtmeleri gerekenleri tam olarak belirtemezler. Bu da projenin gelişme sürecilerini olumsuz etkiler. Bu sorunların kaynağında iletişim eksikliği vardır. XP bu iletişim eksikliğini ortadan kaldırmayı amaçlar. Zaten XP in yapısı buna göre şekillenmiştir. XP de iletişim yüz yüze olmalıdır. İletişim sürekliliği olmalıdır.

Yazılım ekibi ile yazılımı kullanacaklar arasında sıkı bir iletişim bağı olması esastır. Bu sayede sorunlar erken fark edilir. Her hangi bir noktada bilgi alınması gerekiyorsa, kısa bir süre içerisinde bu bilgi elde edilerek, yazılımın gelişme hızı kesilmeden projenin devamı sağlanır.[79]

5.3.2 Basitlik:

XP in ikinci temel taşı basitliktir. Aslında basitlik sağlanması zor olan bir konudur. Basitlik, zorunlu işlerin yapılmasıdır. Gerekli olmayan bir şey varsa, kesinlikle bu konu XP basitlik ilkesi içerisinde olmamalıdır. Dünyadaki en zor şey gelecekte ne gibi ihtiyaçlarla karşılaşacağımızı bilmemektir. Bu nedenden dolayı oluşturulacak yapı, gelecekte ne gibi isterlerin ortaya çıkacağını tam olarak bilmeden oluşur. Buna göre esnek zaman ve maliyeti göz önüne alınmış bir yapı oluşturmak gerekmektedir. XP en iyi şekilde bu basitliği sağlamak için, bu günün ihtiyaçlarını hedef alarak esnek ve basit bir sistem gerçekleştirmeye çalışır.

5.3.3 Geri Besleme:

Sistemlerin geri dönüşümü olması sistemler için paha biçilemez bir fırsat yaratır. Geri dönüşüm sayesinde optimizasyonun oluşmasına engel olan tehlikeler ortadan kaldırılır. Öncelikle programcılar bütün sistemin mantıksal yapısını içeren bölüm testleri oluşturur. Programcılar sistem hakkında somut bilgiler elde ederler. Müşteriler yeni bir “stories” hikâye ile geldikleri zaman, programcılar hemen bu yeni gelen hikâyenin gerçekleşmesi ile ilgili çalışma yapıp bu “stories” e uygun bir çalışma gerçekleştirirler. Bu gerçekleştirilen çalışmalar sayesinde projelerin çoğunda görülen yazılım ekibi ile müşteriler önemli bir sorunları olduğunu ürün teslim tarihinden hemen önce anlarlar. Bu yüzden ürünlerin geliştirilme tarihi ileri bir tarihe sarkar. XP de ise müşterilerle yazılım ekibi belli zamanlarda buluşup, o ana kadar gerçekleştirilen yazılımı çalıştırılıp gelinen noktaya bakarlar. Bu sayede müşteri ile yazılım ekibi arasında sonradan doğabilecek büyük anlaşmazlıklar daha önceden giderilerek müşteri ile yazılım ekibinin ortak bir noktada buluşması sağlanmış olunur.

5.3.4 Cesaret:

XP in dört temel noktasından en zoru cesarettir. Projelerin üzerine yılmadan gidilmesi projelerin geliştirilmesi açısından son derece önemlidir. Cesaretin olmadığı projelerde korku gelişir ve gelişen bu korku projeyi başarısızlığa iter. Yazılım işlerindeki başarısızlık ise; yazılımın çöpe gitmesidir ve maalesef genel duruma da baktığımızda yazılımların büyük bir kısmının çöpe gittiğini görüyoruz. Başarısızlık, genel tabiat içerisinde olan bir durumdur. Başarısızlıktan korkmak yerine başarısızlığı oluşturan nedenler üzerine gitmek yerinde olacaktır. Başarısızlıkla mümkün olunan en kısa sürede karşılaşmak, daha sonra telafi etme şansını arttırır.

XP, başarısızlıktan korkmayı değil en kısa zamanda başarısız olmayı ve onu en kısa zamanda telafi etmeyi önerir. Başarısızlıktan korkmak yazılımcının yavaş hareket etmesine neden olur ve bu da projenin hızının düşmesini sağlar. Sonuçta gelecek olan başarısızlığı ortadan kaldırmaz sadece geciktirir. Proje içerisinde de çıkabilecek herhangi bir mantıksal hata görüldüğünde onu ertelemek yerine üzerine gidilmesi çok daha yerinde olacaktır. Diğer bir cesaret örneği de belli noktalara kadar gelen ama proje gelişimi için sorun yaratan sistemlerden ya da kod parçalarını atarak daha iyi bir sisteme geçme cesaretidir. Bu sayede hem projelerin kaliteleri artar, hem de daha etkin bir yazılım oluşturulması sağlanmış olur. Örnek olarak bir yazılım geliştiriyorsunuz ve bu geliştirilen yazılım içerisinde günden güne problemler artıyor. Burada bir kararın verilmesi gerekiyor. Bu sistemle mi devam edilecektir ve günden güne sistemin kötüye gittiği mi görülecektir. Yoksa, sistemde ileri çapta bir revizyona mı ihtiyaç vardır? Burada XP bire bir dizayn alternatifi sunuyor. Hill-climbing diye adlandırılan bu sistemde ise basit bir dizayn yapılır. Arkasından daha kompleks bir dizayn yapılır. Daha sonra tekrar basit bir dizayn ve daha kompleks bir dizayn olarak bir birini takip eder.Bu sayede proje maliyetini düşürür.Gelişim süresini kısaltır.

5.4 XP in 12 uygulaması

5.4.1 Planlama Oyunu

Başta kaba plandan sonra, kısa süreçler için ayrıntılı plan yapılır. Bu plana göre işler yapılmaya çalışılır. Aksaklık olursa başka bir plana geçilme serbestliği vardır. İşler çok detaylı olarak planlanmaz. Başta yapılan plan detaydan uzaktır. Sadece ilk başta görülen sorunları tam anlamıyla çözmeye yakındır. Yapabileceğimiz ve görebildiğimiz kadarı planlanmaya çalışılır. [80]

5.4.2 Ekipte müşteri

İşleri gerçekleştirmek için müşteri varlığına ihtiyaç duyulur. Müşteri temsilcisi yazılım ekibi ile birlikte yazılımı gerçekleştirme süreci sırasında aynı ortamda bulunur. Bu sayede yazılım gerçekleştirilirken yazılımcının ihtiyaç duydukları bilgilere çok kısa sürede erişilmesi olanağı sağlanır.[81]

5.4.3 Önce Test

Kod yazılmadan önce test programı yazılır. Bu sayede mevcut sorunların daha erkenden ortaya çıkması sağlanarak, daha güvenli bir yazılım gerçekleştirilmeye çalışılır.

5.4.4 En Basit Tasarım

Müşterilerden alınan istekleri minimum şekilde sağlayan bir tasarım ortaya konur. Müşterinin ileride ihtiyacı olacak isterler fazla dikkat edilmeden, sadece o süre içinde oluşmuş isterlere göre bir tasarım yapılır. [82]

5.4.5 Çiftli Programlama

Pairprogramming yapısı ve mantığı ile alışılmamış bir yöntem gibi görülse de günümüzde pek çok şirket pairprogramming kullanmaya başlamıştır. Pairprogramming in tarihi XP ortaya çakmadan öncesine dayanır. Bazı şirketlerde 1980'lerin sonunda pairprogramming uygulanmıştır. Bunun nedeni iş gücünde bir artış olmasından ziyade bilgisayar masraflarının çok yüksek olduğu için bütün şirketlerin her bir programcıya bir bilgisayar alamamasıdır. Daha sonraki yıllarda bilgisayar fiyatlarını ucuzlaması ve değişik yazılım metotojilerinin çıkmasıyla beraber Pairprogramming kavramı yazılım geliştirme literatüründen silinmiştir XP in çıkışıyla birlikte tekrardan gündeme gelen pairprogramming tekrardan yazılım geliştiriciler için bir uygulama geliştirme tekniği olarak karşılarına çıkmıştır.

Geçen süre zarfı içerisinde unutulan bir teknik olduğu için ilk başlarda uygulama geliştiren programcılara zor gelen bir teknik olmuştur. Gerçektende pairprogramming bazı tekniklere bağlı kalınarak uygulanması gereken ve yapılmadığı zaman tam anlamıyla uyulama geliştiricisi için bir kâbusa dönen bir tekniktir. Pairprogrammingle yazılım geliştirmek isteyenler ilk başlarda zorlanabilirler. Farklı konularda uzman olan yazılım geliştiricilerin birbirlerinin söylediklerini tam olarak anlamadıkları için bir arada yazılım geliştirme işinde zorlana bilirler. Bu da yazılım isterlerinin gerçekleştirilememesini ve sürenin uzamasına sebep olur. Zamanın ilerlemesiyle birlikte uzman yazılımcılar birbirlerini daha iyi anlamaya başlamaları ile ilk zamanlardaki isterlerin tam olarak karşılamama olayını çözdüğü gibi işlerin vaktinden önce yapılmasını da sağlayabilmektedirler. Örnek olarak genel planda süresi beş gün olarak verilen bir iş proje planına göre yedi günde bitebilir fakat programcıların bir birlerini tanımasıyla beş gün verilen bir iş planı çok daha kısa bir süre içerisinde gerçekleştirilebilir.

Farklı bilgilere sahip olan programcılar için kendi bilgilerini geliştirme olanağı da sağlar. Örnek olarak T-SQL konusunda iyi olan bir bilgisayar programcısı ile iş zekâsı ve veri modelleme konusunda iyi olan programcılar pairprogramming yaptıkları takdirde kendi bilgilerinin yanı sıra uygulama

geliştirirken diğer bazı tekniklerin nasıl daha efektif olarak kullanılacağını da bir birlerine öğretmiş olurlar.

Program geliştirilme kısmında ise her bir programcı yetenekleri ve bilgileri doğrultusunda projenin gerçekleştirilmesine yardımcı olurlar. Bu da projenin geliştirilme hızını arttıran bir yöntemdir. Çoğu zaman programcılardan biri diğerinden daha hızlı olarak sorunları çözer buda programın yavaşlamasını engeller eğer tekbir programcı o işle uğraşıyor olsaydı ve bir soruna geldiğinde takılsaydı aynı sorunla uğraşan diğer bir programcı olmadığından proje süresi uzayacaktı. Böyle durumlardan kurtulmak içinde pairprogramming kullanmak yararlı olacaktır. Aynı şekilde bir projenin tasarımında ve planlamasının oluşturulmasında da pairprogramming yöntemi kullanılması yararlı olacaktır.

Pairprogramming kullanmanın bir diğer avantajı da farklı bakış açılarının proje üzerinde odaklanması gerçekleştirilecek projede ileriki aşamada ortaya çıkacak bazı sorunların ortaya çıkmadan proje planlama aşamasında çözülmesini sağlar. Bu avantajı ile de kodu yamama işlerinden yazılımcıyı kurtarır.

5.4.5.1 Hangi alanlarda pairprogramming kullanılması daha yararlı olur?

- Yazılımdaki hataların giderimi (bugfixes)
- Yazılım güvenliği (job security)
- Modül birleşim kısımlarında kullanımı daha verimli olacaktır.

Ama projenin tamamı içinde pair programming kullanılabilir.

Pairprogramming eğer doğru bir şekilde kullanılmaz ise üretim hızını önemli bir ölçüde azaltabilir. Pairprogramming yöntemini kullanacak programcılar arasındaki uyum beklenilen zaman dilimi içerisinde gerçekleşmez ise bu üretim sorunu oluşacağı anlamına gelir. Ama bu sorun ortaya çıkmaz ise programcılar pairprogramming yöntemini kullanmaları yararlı olacaktır.

5.4.5.2 Yeni pairprogramming teknikleri

Yeni ortaya çıkan pairprogramming tekniklerinde ise şirketler yazılım geliştirme guruplarını üçerli gruplara ayrılırlar. Bu guruplara önce insan mühendisliği teknikleriyle kişilerin arasındaki uyum ölçülmeye çalışılır. Bu gruplar oluşturulduktan sonra üç kişinin yapılacakları işleri belirlenir hangi işlerin tek hangi işlerin pairprogramming yöntemine göre yapılacağı belirlenir. Bir programcı ayrı diğer iki programcı ortak olarak çalışır. Gereken yerlerde bu programcılar dönüşümlü olarak pairprogramming yapalar.

5.4.6 Sürekli Tümleştirme

Her küçük kod parçası tümleştirilir. Kod bir araya getirilerek modül uyuşmamasından dolayı olabilecek entegrasyon hataları giderilir. Haftada bir ya da iki kez kod entegrasyonu işi gerçekleştirilir.

5.4.7 Kısa sürümler

Müşteriye kısa ararlıklarla programın o zaman ki sürümü verilir. Bu sayede müşteriye yazılımın ne durumda olduğu gösterilir. Müşteri yazılımı çalıştığı kadarıyla, görmesi müşteriyi memnun eder. XP avantajı diğer metot ve motetolojilere göre daha erken bir süre içerisinde kullanıcıya çalışabilir bir sürüm ulaştırma imkânını vermesidir. Bu sayede kullanıcı programın tamamlanan kısımlarına daha erken safhada bakma imkânı kazanır. Bu sayesinde oluşan hatalar daha kısa bir sürede görülmesi sağlanır. Hataları ne kadar erken görebiliyorsak bizim açımızdan programın gerçekleştirme işini de o denli kolaylaşır program maliyetinde de düşüş sağlanır.

5.4.8 Yeniden yapılandırma

Kod ve tasarım sürekli gözden geçirilir, bunun iki farklı amacı vardır. Birincisi gerçekleştirilen yazılımı daha kolay ve basit bir biçimde yapılabilir. İkinci amaçta müşteriden gelen yeni istekleri mevcut şart ve durumlara bakarak nasıl geliştire biliriz sorusunun cevabı aranır. Proje içerisindeki ana fikir müşterinin memluniyetidir.Bu iki amaç için yeniden yapılandırma kullanılır.

5.4.9 Ortak sahiplenme

Kod, kodu üreten bütün yazılımcıların malıdır. Farklı kısımları üreten kimseler diğer kısımlarla ilgili değişiklik yapmazlar diyen bir fikir yersizdir. Belli kurallar çerçevesinde başkasının ürettiği kodlara da erişilip mevcut yazılımı daha iyiye götürme adına değişiklik yapılabilir. Extreme Programming yöntemimi kullanan takım üyeleri basit bir planlama ve izleme formu kullanırlar. Bu sayede proje bittiğinde nelerin olacağına karar verirler. Takımın bütün üyeleri yapılan işe odaklanarak, ortaya çıkacak ürünün kalitesini arttırmaya çalışır. Yapılan bütün çalışmalar sonucunda modüller birleştirilir ve daha önceden tanımlanan testlerden geçirilir.[83]

5.4.10 Metaphor

Gerçekleştirilecek yazılımda sistemler birbirlerine benzetilerek yazılım yapılmaya çalışılır.Örneğin gerçekleştirilecek yazılımı parçalara ayrılır. Her bir parçası bir başka sistemle benzeştirilir. Mesela bir resimle yazılımın her bir parçası bittiğinde o resme ayıt bir parçada bitmiştir denilmektedir. Yapboz da olduğu gibi yavaş yerine oturan parçalarla resim oluşmaya başlar, resim bittiğinde oluşturulan yazılımda bitmiştir.

5.4.11 Kodlama standardı

Kodlama standardının tek sitil olmasında yarar vardır. Gerçekleştirilecek yazılımda oluşturulacak standart örneğin editlere verilecek isimler, formların adları, değişken isimleri, fonksiyonlarının tutulacakları yerler vb konuların tekbir standartta olması kodlamanın anlaşılırlığı ve sadeliği açısından önemlidir. Önceden yapılan tanımlara uygun bir şekilde belirlenmiş bir standartla yapılan yazılım, farklı kodlama teknikleri kullanmadığından diğer yazılımcılar içinde anlaşılması daha kolay bir yapı barındıracaktır. Bu sayede kodlama sırasında oluşabilecek karmaşıklıkları da engellemiş olur.

5.4.12 Kırk saat haftada

Çalışma verimliliği açısından hafta 40 saatlik bir çalışma süresi ayrılır. İşler bu 40 saatlik dilimde bitirilir. Hafta içi her gün 8 saatlik bir çalışma süresi ile işler gerçekleştirilir.40 saat haftada uygulamasında fazla mesai kavramı yoktur. Sadece haftada bir kez fazla mesai yapılabilir. Bu uygulamaya göre 40 saatten fazla çalışılsa bile verimde yeteri kadar artış olmaz. Hatta yapılan hata sayısında artış meydana gelme riski de artar. Eğer 40 saatlik çalışma süresi yetmiyorsa o zaman sistem iyi bir şekilde düzenlenmiyor demektir. Sistemi yeniden ele alıp iş akışlarına tekrardan bakılması gerekmektedir.

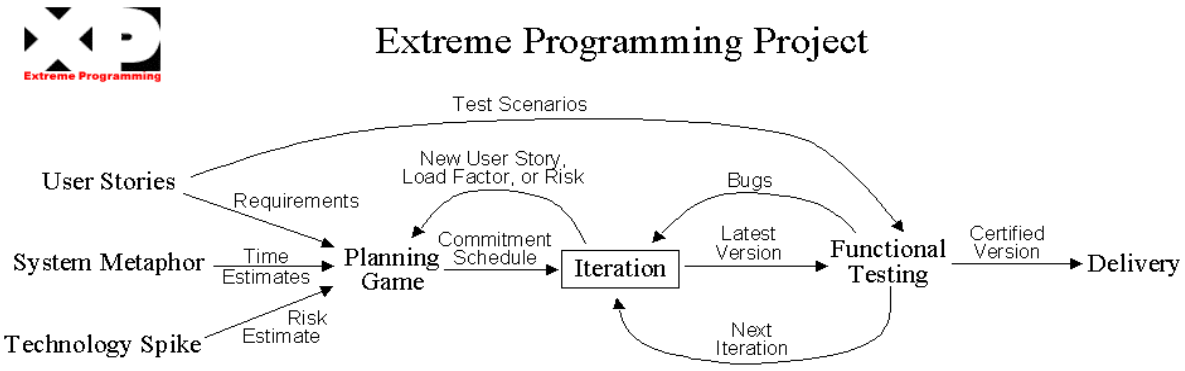
5.5 Disiplinli metotlarla XP karşılaştırılması

Tablo 5.3 Çevik Metotlarla Disiplinli Metotların Karşılaştırılması

Çevik Metotlar	**Disiplinli**
Küçük projeler için uygundur.	Büyük Projeler.
6 ay-12 ay arası işlerde küçük ekip 10-15	Proje süresi > 12 ay
Bu tip projeler personel değişimden etkilenir.	Personel Değişiminden Etkilenme daha azdır.
Daha deneyimli tecrübeli yazılımcı ihtiyacı vardır.	Ortalama yazılımcılar ile işler sürdürüle bilir.
Temel Eğitim Önemlidir	Süreçler için verilecek eğitim önemlidir
Değişiklik Kolaydır.	Değişiklikler Zordur.
Müşteri esnekse, ihtiyaçları değişiyorsa ve yazılımcı gurupla iş birlikçi bir tutum içerisindeyse kullanılabilinir.	Müşteri katı ise isteklerinde fazla değişim gözlenmiyorsa.
Proje bakış açısı aşağıdan yukarı	Proje bakış açısı yukarıdan aşağı

5.6 Çevik Projelere Başlarken

Çevik projeler kullanıcı senaryolarıyla başlar. Birim sürede bitecek şekilde kullanıcının anlattığı hikâye alınır parçalara ayrılır. Yapılacak işte kullanıcının hangi bilgileri sisteme gireceği ve hangi bilgilere ulaşacağı tespit edilir.(User Case).Her bir modül ile ilgili bilgi durumu 3*7 lik kartlara yazılır. Modül müşterinin ihtiyaçlarını karşılar hala gelince karta bir check atılır. Her bir kart basit olmalı müşterinin isterleri sade bir dil ile anlatılmalıdır. [85]



Şekil 5.2 Proje Çevrimi

5.7 Testlere Öncelik

Klasik yazılım gelişim sistemlerinde analiz dokümanları hazırlanır, dokümanların hazırlanma aşaması bittikten sonra kodlamaya geçilir. Kodlama tamamlandıktan sonra da klasik testler yapılıyor, dönen hatalar veya eksiklikler sebebiyle, sürekli test yapılarak müşterinin istekleri karşılanmaya çalışılıyordu.

Extreme Programming de ise klasik yöntemlerden farklı olarak bir test kodu yazılmaktadır. Test Kodu, bir modülün doğru çalıştırıldığından emin olmak için programcılar tarafından oluşturulan bir koddur. Test kodunun amacı, çalışan programlar üzerinde yapılan geliştirme çabalarının programın genel yapısına bir yanlışlığa sebep verip vermediğini anlamak için oluşturulur. Bu sayede verilen girdilere uygun çıktı alınıp alınmayacağında anlaşılmış olur. Burada kullanıcılar ana programı kullanırlar. Test programı ise programcılar tarafından kullanılır. Bu kullanım ya belli zamanlarda bu test kodu çalıştırılarak ya da otomatik olarak kodun kendi kendini çağırmasıyla sağlanabilir. Testler iki farklı şekilde olabilir; sonuca odaklı testler ya da program içerisinde iç bilgilerin üretildiği kısımda oluşan testler.

Test programı yazılması için öncelikle ana programa girecek girdiler ve çıktılar belirlenir. Her test kısmı için bir kontrol listesi hazırlanır. Burada ana programa girilen girdilerle test programının girdileri tam olarak aynı girdiler değildir. Ana programa yapılan girdiler klavyeden yapılan girdiler iken test programına yapılan girdiler, test programı oluşturulmadan ortaya konulan her türlü test senaryosuna uygun verilerdir. Burada test senaryosuna ait verilerin girileceği bir

veritabanı kullanmak çok yararlı olacaktır Ayni şekilde test programına ait çıktılarında bir veritabanında tutulması çok yararlı olacaktır.[85]

Burada çıktılar programın kullanıcıdan beklediği tüm verilerdir. Test programı tarafından oluşturulan test verileri ile ana programdan gelen çıktılar karşılaştırılmasıyla oluşan verilerde test programının kontrol listesidir. Bu listede hatalı olan bilgiler ile hatalı olmayan bilgiler listelenmiştir. Bu sayede hataya sebep verebilecek kod blokları da kontrol edilmiş olunur.

Burada en önemli konulardan bir diğeri de girdilerin programa nasıl aktarılacağıdır.Burada programcıların hazırladıkları test senaryoları ile yapılacak girişler ile veriler oluşturulmakta ve oluşturulan çıktılar test için yaratılan yeni bir veritabanına kaydedilmektedir. Test programının sonunda verilerde tutarsızlık veya hata olduğunda ise kontrol listesin hata kodu olan değer yazılmaktadır. Bu verilere bakarak programcı hangi noktada hata olduğunu anlayabilir. Test programı yazmanın bir diğer avantajı da, hata kodlarını çeşitli kategorilere ayırarak yazılım karmaşıklığı azaltmakta yardımcı olmasıdır. Böylelikle programcı test programını çalıştırıp sonucunda aldığı verilerden hatalar varsa bu hataların nerelerden kaynaklandığını öğrenebilir. Bu öğrenme işi kategorilere ayrılan hata kodlarından alınan verilerle sağlanır. Karmaşıklığı azaltmak için izlenecek bir diğer yol ise ayrı bilgi girişleri için ayrı test kodları yazılmasıdır. Eğer böyle bir yöntem izlenirse karmaşıklık oranı biraz daha azaltılmış olacaktır. Tabi ki böyle bir tercih yapıldığında her bir bölüm için farklı veri gruplara veya test gruplarına ihtiyaç vardır. Yardım için hazırlanan fonksiyonlar ve ekrana bilgi listeleme fonksiyonları içinde test veri tabanı hazırlanmış olması gerekir. Aynı şekilde boş bir veri tabanının da çıktılar için oluşturulması gerektir.

Test veritabanlarında en çok görülen rahatsızlık ise liste alma sırasında görülmüştür. Bu liste işlemlerinde oluşan detaylarla doğru orantılı olarak karmaşıklaşmanın da artma neden olmaktadır. Bu karmaşıklaşmadan kurtulmak için aynen veri girişlerinde yapıldığı gibi bu işlerin belli kategorilere ayırarak yapılması çok daha yerinde olacaktır. Bu işler yapıldığında testler sırasında oluşan hatalara

ulaşmak daha kolay olduğu gibi hatalarında nerelerden kaynaklandığını bulmakta çok daha kolay olmaktadır.

5.7.1 Test Kodu ve Ana Yazılım

Yapılan ana programda hataların nerelerden kaynaklandığını bulmak uzun bir vakit gerektirir. Bu nedenden dolayı test kodunu doğru bir şekilde yazmak çok önemlidir. Ama test kodunun ana veri tabanı kullanmasından ziyade kendi veri tabanı kullanması gerekir. Bunun asıl amacı test süresini kısaltmaktır. Eğer yeni veriler üzerinde bir işlem yapılmak isteniyorsa ve bizim test veri tabanımız eski ise bu verilerin test için oluşturulan özel veritabanına girilmesi yeterli olacaktır.

Program üzerinde bir değişiklik yapıldığı vakit ise programın çalışmasının ardından test programı çalıştırılarak girilen veriler üzerinde yeni programın bir hataya sebep verip vermediğini kontrol edilecektir. Kontrol için kullanılan listelerle inceledikten sonra bir hata bulunamadıysa bu yeni değişiklik onaylanır. Bu değişikliği ilgilendiren bir başka veri değişikliği varsa o veri değişikliği veritabanında yapılır. Aynı şekilde test veritabanına da bu bilgi değişikliği yapılır. Arkasından da ihtiyaç duyulursa test programına ilişkin güncellemeler yapılır. Burada unutulmaması gereken test programının hiçbir zaman statik olmadığıdır. Yeni gelişen durumlar varsa kendi yapısını değiştirmesi gereken yerlerde değiştirir.

5.7.1.1 Test Kodu Yapılırken Dikkat Edilmesi Gereken Kurallar:

- Test için gereken verilerin gelişi güzel seçilmesinden ziyade veri seçimi işine dikkat edilip veriler ona göre seçilmelidir.
- Test için kullanılacak veriler başlangıçtaki girdi ve çıktıların neler olduğu hangi bilgi blokları içinde tutulacağı ve bunların nasıl test edileceğine dikkat edilmelidir.
- Asıl program değişiklikleri ile paralel olarak test programı değişiklikleri gözden geçirilmelidir.

5.7.1.2 Test Kodu Yazmanın Avantajları:

1-) Farklı veriler kullanarak değişik koşulların testi yapılabilmektedir.
2-) Test kodu yazılırken değişik test senaryoları kullanarak yazılacak ana programın nasıl şekilleneceği ile ilgili bazı bilgiler edinile bilir.
3-) Ana programda ortaya çıkacak hataların sayısı azaltılmış olacaktır.
4-) Testler istenildiği zaman otomatik yaparak süreçlerin doğru yürütülüp yürütülmediği anlaşılabilir.
5-) Kişisel test senaryolarıyla kaybedilen zaman ve gözden kaçan test senaryoları çok daha iyi bir şekilde uygulanabilir.
6-) Ana programdan önce test programına başlandığı için ana programla ilgili eksik kalmış yerler daha önceden fark edilebilir.
7-) Yapılan değişikliklerin veriler üzerinde herhangi bir etkisi olup olmadığı saptanabilir.

Sonuç olarak ana programı yazmadan test programını yazmak ile programlarda sonradan çıkacak hataların azaltılmasına sağlayan son derece etkili bir yöntem olduğu görülmüştür. İlk başta yeni bir yöntem olmasından dolayı kullanımı zor bir yöntem olsa da ileriki safhalarda yazılımların daha kaliteli yazılımların oluşmasını sağlayacağı görülmüştür.

5.8 XP ile RUP

5.8.1 RUP (Hızlı Birleştirici Süreç)

Rational firması tarafından ortaya çıkarılan ve yazılımda süreç geliştirmeyi çok detaylı bir şekilde ele alan bir yazılım geliştirme modeli olan RUP zengin tümleşik araçları ve hazır belge şablonları ile yazılım ekibinin üretkenliğini ve hızını arttırmak için oluşturulmuştur. RUP'un esnek yapısı ile firmalara özgün bir yapı oluşturma avantajı sunar. RUP organizasyonlar için detaylı bir yapıya sahiptir. RUP içerisinde çeşitli aktivite gurupları vardır. Bunlar gereksinimler, analiz, tasarım, gerçekleştirme ve testtir. Yazılım aşamaları ise dört tanedir. Başlangıç (Inception), düzenleme (Elaboration), oluşturma (Construction) ve geçiş (Transition). RUP yazılımı oluştururken bu dört aşamayı birer durak noktası gibi değerlendirir. Her aşamanın sonucunda bu aşamada istenilenler ne ölçüde gerçekleştirildi diye bakılarak bir sonraki aşamaya geçilir. [86]

Bir Proje İçerisindeki Yazılım Aşamalarının Projeye Oranı:
%10'unun Başlangıç
%30'unun Düzenleme
%50'sinin Oluşturma
%10'unun Geçiş

Başlangıç Aşaması: Tahminler ortaya konulur temel gereksinimlerin ne olduğu belirlenir. Proje kabataslak bir yapısı ortaya konulmaya çalışılır. Vizyon belirlenir. Kullanım senaryolarına başlanır, projede kullanılacak terimler oluşturulur, maliyetler ve fayda analizleri belirlenir.
Düzenleme Aşaması: Daha detaylı tahminler yapılarak mimarı oturtulur, riskler belirlenir ve bu risklerle karşılaşılması durumunda yapılacaklar ortaya konur. Kullanım senaryoları ve bu yazılımı kullanacak kişilerin kullanıcı tanımları büyük ölçüde belirlenir, tasarım modeli ortaya konur, veri gerçekleştirme test modelleri kullanıcı ara yüzleri hazırlanır.

Oluşturma Aşaması: Yazılımın oluşturulduğu kısımdır. Bu aşamada varsa hatalar belirlenir. Kaynak kodlar, birim testleri ve projenin genel testleri yapılır. Test senaryoları oluşturulur, test kodları gerçekleştirilir, müşteriye verilecek ürünün, kullanıcı el kitabı oluşturulur.

Geçiş Aşaması: Yazılımın teslim edildiği aşamadır. Bu aşamada ayrıca müşteriden gelen yeni isterler doğrultusunda yazılım geliştirme işleri de yapılır. Müşteri memnuniyeti hesaplanır. Bu aşamalar bittikten sonra müşteri memnuniyeti belirlenmeli müşterilerinin isterleri ne ölçüde karşılandığının objektif bir araştırılması yapılmalıdır.

5.8.2 XP İle RUP Karşılaştırmaları

XP & RUP birbirleriyle bir arada kullanılıp kullanılamayacağına bakmadan önce benzerlikleri ve farklılıklarına bakmak gerekmektedir. Proje büyüklüğü olarak baktığımızda RUP orta ve büyük projelere uygunken XP daha çok küçük ve orta büyüklükteki projelere uygulanmaktadır. Yapı içerisinde kendini yenileme işine ise RUP & XP de özyinelemenin olduğunu görüyoruz fakat XP de öz yineleme daha fazladır. XP de özyineleme işleri daha kısa süreler içerisinde yapılabilir. Kullanım senaryoları olarak baktığımızda her ikisinin de kullanıcı senaryolarına göre bir yapıları vardır. RUP un ilk aşaması XP in kullanıcı senaryoları aşamasına RUP un ikinci aşamasında XP in oyunu planlama aşamasına benzemektedir. XP ve RUP da basitliğe önem vermektedirler. Farklılıklara baktığımızda ise RUP XP den daha fazla risk azaltma işi ile ilgilenir. XP daha uzman eleman istisnamı gerektirmektedir. XP kullanan firmalardaki elemanlarının sorumlulukları daha fazladır. XP çok fazla belge kullanmaz RUP ise çok fazla belgeleme kullanır. RUP XP den daha fazla mimariye önem verir. XP de yazılımcı, müşteri, koç ve yönetici rolleri vardır. RUP ta ise programcı ve müşteri tanımları aynen XP de olduğu gibi vardır XP den ayrı olarak rol tasarımcısı ve proje yöneticisi tanımları da vardır.[86]

5.8.3XP İle RUP Bir Arada Kullanılabilir mi?

Genel kullanım şekli büyük projelerde RUP küçük projelerde ise XP kullanımı şeklindedir. Yazılım ortamına kattığı yeni tekniklerle XP (uç programcılık) yazılım üretimi açısından kullanılabilecek bir tekniktir. Bu nedenle XP ile RUP bir arada kullanabilirsek daha efektif yazılımlar oluşturma şansı elde edilebilir. Projelerde önce müşteri ile çalışılarak kullanıcı senaryoları oluşturulur daha sonra XP de olduğu gibi müşteri ekibe dâhil edilerek bu kullanım öyküleri detaylandırılır. Burada RUP un başlangıç tanımları kullanılabilir. Tasarım kısmında sistem mimarisi oluşturulabilir.Oluşturulan bu sistem mimarisinin detaylı bir tasarımını XP ile yapılabilinir. Test aşamasında ise XP in birim testleri kullanılarak test kısmı yapılabilinir. Geliştirilme kısmında ise çiftli programlama, sürekli tümleştirme gibi XP özellikleri kullanılarak geliştirme kısmı gerçekleştirilir. Kısacası XP ve RUP karmaşık sistemlerin olduğu durumlarda titizlikle kullanıldığı takdirde başarılı bir netice alına bilinir.

5.9 XP ile Risk Yönetimi

Projeler geliştirilirken var olan risklerden kaçınılamaz. Riskler proje içerisinde hep vardır. [87]

Proje içerisinde karşılanacak en önemli riskler ise

- Gecikmeler
- Projenin iptali
- Yama tutmayan sistemler
- Hata oranı
- Yanlış anlaşmalar
- İş dünyasındaki birçok değişiklikler
- İhtiyaç duyulmayan birçok özellik
- İşten çıkmalar

Bu Risklerle mücadele etmek için XP:

Geçilmeler = Kısa sürüm zamanları
Projenin iptal edilmesi = İş acısından en önemli süreçlere önem verilmesi.
Yama tutmayan sistemler = Anlaşılır test senaryoları (otomatik test)
Hata oranları = Testler
Yanlış anlaşmalar = Müşterinin takımın ayrılmaz bir üyesi olması sayesinde yanlış anlamaların büyük bir kısmı önlenir.
İş dünyasındaki değişiklikler = Kısa sürüm zamanları
İhtiyaç duyulmayan birçok özellik = Sadece en öncelikli işlerin yapılması
İşten çıkmalar = Çift programcı

5.9.1 XP'in korkmadıkları

- Kodlama
- Değişen fikirler
- Geleceği bilmeden yol alabilmek
- Diğer insanlara güvenmek
- Çalışan bir sistemin yapısının değişmesi
- Testler yazılması

XP de müşteri ile yazılımcı ilişkileri sürekli olursa yanlış olayların olması engellenir.

5.10 XP'deki Roller

- Müşteri
- Yazılım Geliştirme
- Test Sorumlusu
- Ölçüm sorunlusu
- Koç
- Danışman
- Yönetim

Müşteri: İş tanımını yapar. Neyin yapılacağına ve bu yapılacak konunun ne kadar önemli olduğuna karar verir. Sistemin oluşabilmesi için gerekli olan kabul testlerini tanımlarlar.

Yazılım geliştirici: İletişim ile görevlidir. Olayları nasıl daha basit bir şekilde gerçekleştirilir sorusu ile ilgilenir. Kodlama, test, sistem entegrasyonu müşteri senaryolarının özelliklerinin ve zorluklarının tespiti.

Test Sorumlusu: Ekip büyükse 1 kişi bu işlerle uğraşabilir. Eğer yeterli ekip yoksa ekip içerisinden pairprogramming ile ilgilenen bir kişi bu işi gerçekleştire bilir. Müşterilere test yazma konusunda yardımcı olur. Test araçlarının fonksiyonel olarak doğru çalışıp çalışmadığına bakar. Bütün testlerin düzgün çalışmaları ile sorumludur. Geçen ve geçmeyen testlerin raporlarını da tutar.

Ölçüm Sorumlusu: Gerçekleşen değerleri bir araya getirilmesi. Gerçekleşen değerler ile önceden tespit edilene değerleri karşılaştırılması. Gidişatın tespit edilmesini sağlar.

Koç: Süreç uzamanıdır. Proje içerisinde çıkabilecek sorunları önceden tespit edilmesini sağlamaya çalışır. Sorunlara müdahale zamanlarını belirler.

Danışman: Teknik özel sorunları çözmeye çalışır. Çözümü takım üyeleri ile birlikte bulmaya çalışır.

Yönetici: Müşteri tarafından verilmesi gereken kararlara ya da programcı tarafından verilmesi gereken kararlara karışmaz. Müşteri ile programcıların yazılım içerisinde bir arada uyum içerisinde çalışmalarını sağlayacak şartların oluşmasını sağlar. Toplanacak ölçütlere karar verir. Ödüllendirmeleri yapar. Gereksinim ihtiyaçlarını saptar.[88]

5.10.1 XP'nin Müşteri hakları

Genel planı ve maliyetleri bilmek. İşlerin ne zaman biteceğini ve bu işlerin sonucunda çıkacak maliyeti bilmek. Çalışan programı görmek. Otomatik test programlarının sonuçlarını görmek.

Yapılacak işleri yeniden tarif edebilmek. Yaşanan gecikmeler var ise bunları mümkün olan en kısa süre içerisinde bilmek. XP in müşteri haklarındandır.[89]

5.10.2 XP'deki Programcı hakları

Gereksinimleri bilmek ve nelerin önce yapılacağından haberdar olmak. Yardım alabileceği bir ortamda çalışmak ise programcıların haklarıdır.

5.10.3 XP ne zaman uygulanmaz ?

XP her proje her müşteri ya da her şirket için uygun bir model değildir. Şirket yapısı, müşteri kültürü, yönetimsel yapı, çalışma ortamı ve proje ekibinin büyüklüğü 10 kişiden fazla olduğu durumlarla XP kullanımı uygun değildir.

6. CMMI İle Extreme Programming Değerlendirilmesi

1970'lerde başlayan kaliteli yazılım üretme çabaları sonucunda en iyi uygulamalara bakılarak ortaya çıkan ve daha önceki bölümlerde anlatılan CMMI ile hızlı ve esnek yazılım geliştirmeyi amaçlayan XP birbirleri ile çoğu zaman karşılaştırılmıştır. CMMI'da XP'de yazılım üretme işi ile ilgilenmektedirler, ama olaya bakış açıları değerlendirilmeleri bir birinden faklıdır.

CMMI bir değerlendirme yaparak sonuca gider, işlerin ne kadar iyi yapıldığına bakarak değerlendirmeyi gerçekleştir. İşlerin nasıl yapılması gereğini ortaya koymaz.. Şirketlerin sağlık durumlarına bakan bir check up gibidir. XP ise, çalışma ortamının nasıl olacağı ile ilgilenir. İşlerin nasıl yapılacağı ile ilgilenir. Bir check up tan ziyade, insanın sağlıklı olması için yapması gerekenleri söyler. Örnek olarak CMMI bir 100 metre koşucusunun 100m yi 10 saniyenin altında koşması gerektiğini söyler. XP ise 10 saniyenin altında koşmak için her gün şu şekilde çalışmalısın ve bu pratikleri yapmalısın der. Yani CMMI bir teşhis XP ise tedavidir. Bu nedenle bir birleri ile karşılaştırılmaları yerine avantajları bir araya getirilerek yazılı üretme işi nasıl daha iyi olacaktır sorusu üzerinde durulmalıdır.

Eğer XP ile CMMI birbiri ile karşılaştırmamız gereği ortaya çıkmış olsaydı. Aşağıdaki gibi bir tablo ile karşılaşırdık. CMMI ile XP karşılaştırdığımız zaman Tablo 6.1 deki gibi bir sonuçla karşılaşırız.

Tablo 6.1 XP CMMI karşılaştırılması

XP	CMMI
İnsan Temellidir	Süreç Temellidir
Değişikliklere Açıktır	Değişiklikler kontrol altındadır.
12 Pratiği vardır	368 tane anahtar pratiği vardır.
Küçük yazılım ekipleri için uygundur	Genelde büyük yazılım ekiplerince tercih edilir.
Uzman bir yazılım ekibi ister	Yüksek kalitede yazılımcıya XP kadar ihtiyaç duymaz.
Belgeleme azdır	Belgelemeye fazla önem verilir.
Projelere bakış açısı aşağıdan yukarıdır.	Projelere bakış açısı yukarıdan aşağıdır.

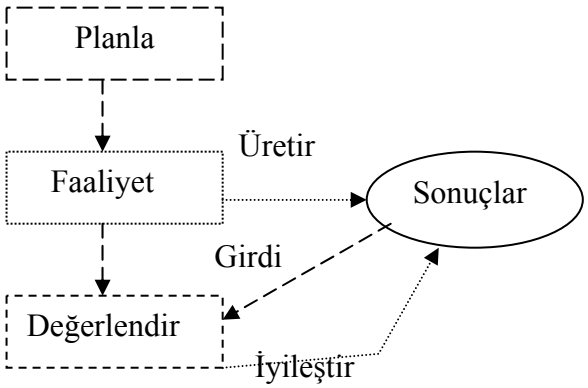
Sonuç olarak, bizim asıl ilgilendiğimiz konu olan ve tezinde amacı teşkil eden; CMMI'ın beklentilerini XP kendi yaşam döngüsü içerisinde nasıl karşılayacaktır?

Aşağıda CMMI 2.seviye süreçlerin isterlerinin XP tarafından nasıl karşılandıkları sorusuna cevap verilmeye çalışılmıştır.

6.1 CMMI 2.Seviye Süreçleri İle XP Pratiklerinin Karşılaştırılması

CMMI 2.seviye süreçlerinin XP ile nasıl karşılaştırılacağına bakmadan önce CMMI 2.seviyenin ne anlama geldiğini bir hatırlarsak daha kolay şekilde bu karşılaştırmayı gerçekleştirebiliriz. CMMI 2.seviye süreclerinde, doğru hareket yaptığından emin olmak önemlidir.Bunun için plan yapma gerekliliği vardır.Yapılan plan sürekli olarak izlenir ve geliştirilir.CMMI 2.seviyedeharekete geçmeden düşünülmesi gerekliliği vardır. İşlemler gerçekleştirilmesi sonucu faaliyet değerlendirmesi yapılır. Temel yapılması gereken testler yapalır.

Aşağıdaki şekilde gösterildiği gibi CMMI 2 seviyesinde plan ve değerlendirme vardır. Temel faaliyetler belirlenir.



Şekil6.1 CMMI Seviye 2

Tablo 6.2 CMMI2.Seviye

Seviye	PY	SY	D	Müh	Süreç Alanı	Ö H S	Ö P S	G H	GP[2]
2				Tml	Gereksinim Yönetimi	1	5	2	12
2	Tml				Proje Planlama	3	14	2	12
2	Tml				Proje İzleme ve Takip	2	10	2	12
2	Tml				Tedarikçi Sözleşme Yönetimi	2	7	2	12
2			Tml		Konfigürasyon Yönetimi	3	7	2	12
2			Tml		Ürün ve Süreç Kalite Güvencesi	2	4	2	12
2			Tml		Ölçme ve Değerlendireme	2	8	2	12

Yukarıdaki tabloda gösterildiği gibi CMMI 2. seviyede süreç alanlarının dördüncü ve beşinci seviyelerde olduğu gibi ileri düzeyde değil de temel bazda ele alındığını görüyoruz. Bu yapı da XP kullanılarak CMMI'ın isteklerini gerçekleştire bileceğimiz yönündeki düşüncemizi güçlenmektedir. XP'in en önemli özelliği olan basitlik ilkesinde olduğu gibi aşırı detaylardan uzak olan CMMI 2. seviye ile birlikte XP kullanılabilir mi? Böylesine bir yapı kullanarak CMMI 2. seviye bir şirket

[2] Kısaltmaların Açılımları:
PY:Proje Yönetimi , SY:Sayılarla Yönetilen , D:Destek , Müh:Mühendislik , ÖHS:Özel Hedef Sayısı, ÖPS:Özel Pratik Sayısı ,GH:Genel Hedef, GP:Genel Pratikler , Tml:Temel

olunabilir mi? Şimdi CMMI 2.seviye süreç alanlarının, XP'nin özel pratikleri ve yaşam döngüsünün özellikleri sayesinde karşılayıp karşılamadığına bakalım. Burada amacımız, CMMI 2.seviyesinin isteklerinin XP tarafından karşılanabileceğinin mümkün olup olmadığına bakalım.

6.1.1 Gereksinim yönetimi:

Asıl amacı projeyi ortaya çıkaracak ürünleri ve ürünleri oluşturacak yapıları yönetmektir. Ayrıca proje çerçevesinde ihtiyaç duyulan bileşenler arasındaki tutarsızlıkları ortadan kaldırmayı da amaçlar. XP gereksinim yönetimi için kısa zamanda oluşturulan sürümler, takımda müşteriden bir kişinin bulunması gibi özellikleri kullanarak gereksinim yöntemi sürecinin ihtiyaçlarını karşılamaya çalışır.

ÖU1.1: Ortak bir aklın gereksinimleri belirlemesi = XP gereksinim belirleme işini müşteri ile birlikte yapılmaya çalışılır. Bu sayede ortak bir akıl belirlenir. Ekipte müşteride olacağından ihtiyaçlar daha hızlı bir şekilde yazılımcıya ulaştırılır.

ÖU1.2: Taahhütleri alarak gereksinimlerin karşılanması = XP'de müşterilerden alınan kartlar taahhüttür. Müşteri ile yazılımcı arasında kartlar sayesinde gizli bir taahhüt vardır.

ÖU1.3: Gereksinler ve gereksinimler üzerindeki yönetimler = XP'in yapısı gereği müşteriler isteklerini zamanın getirdiği şartlar doğrultusunda belli oranlarda değiştire bilir. Değişimler olsa bile XP proje yaşam çevrimi sayesinde istekleri karşılamaya çalışır. Bu gereksinimler üzerindeki yönetimde ekipteki koçun rolü çok önemlidir.

ÖU1.4: Gereksinimlerin müşteri ve yazılımcı tarafından kontrolünün sağlanması = XP'de müşteri ile birlikte yazılımcı aynı ortamda olduklarından gelişmeleri bir arada değerlendirebilir.

ÖU1.5:Eldeki durumla gereksinimlerin karşılanması = XP yaptığı otomatik test programları ve kısa sürelerde ortaya çıkan küçük sürümlerle eldeki durumu değerlendirir.

6.1.2 Proje planlama:

Proje planlamanın amacı, proje faaliyetlerini tanımlayan planı gerçekleştirmek ve bu plan doğrultusunda hareket edilmesini sağlanmaktadır. Temel faaliyetleri ise proje planını oluşturmak, paydaşlarla gerekli iletişimin kurulmasını sağlamak, plana taahhütlerin alınmasını gerçekleştirmektedir. Genel tanımı ile planın güncel bir şekilde tutulmasını amaçlar. XP pratikleri de bu amaç doğrultusunda neredeyse proje planlamasının bütün özel uygulamalarını tam olarak karşılar.

XP işin başında mevcut duruma göre bir plan yapılır. Ama unutulmaması gereken bir nokta zaman içersinde gelen değişikliklere göre planın değişebileceğidir. XP buna göre basit, kolayca değiştirilebilir bir yapı kurmak için çalışır .Planlar yapılır. XP proje planlama süreci karşılaştırılması için XP'in pratiklerinden biri olan küçük sürümler özelliğinden yararlanır. Bu sayede projenin yapısının güncelliği korunur. Yazılım takımı belirlenir: yönetici, koç, test, ölçüm sorumluları ve danışman oluşan takım, planın güncelliğini korumasından sorumludur. Plan yazılımın bütün süreci boyunca kontrolden geçirilir Metafor ile de yapılacak projenin genelinde oluşabilecek kopuklukların olmaması sağlanır. Ayrıca zamanlama kutuları ile işlerin ne kadar sürede gerçekleşeceğinin daha önceden hesabı yapılması da sağlayarak proje bütünlüğü korunduğu gibi yazılımın geliştirilme ortamının kalitesini de yükseltmeye çalışır.

ÖU1.1 Proje çerçevesinin belirlenmesi = Proje kapsamı belirlenmesinde XP müşteri kartlarından yararlanır. Müşteri kartları sayesinde proje çerçevesi belirlenmeye çalışır. XP'deki çalışma ortamı da bu proje çerçevesinin belirlenmesinde yardımcı olur. Ayrıca planlama oyunu kullanılarak proje çerçevesi tam olarak belirlenmeye çalışılır.

ÖU1.2 İş ürünleri ve görev özellikleri için kestirimlerin oluşturulması = XP bu işi kendi yazılım oluşturma işleri için geliştirilen teknik kartları kullanarak yapmaya çalışır. Teknik kartların üzerlerinde tarih alanı kısmı ve tahmini süre kısımları vardır. Buradaki kısımlar kullanılarak kestirim işleri gerçekleşir. Kestirim sürelerini projede görevli olan koçlar tarafından belirlenir.

ÖU1.3 Proje hayat döngüsünün tanımlanması= Proje hayat döngüsü tanımlanması kısmını XP kendi hayat döngüsü modelini kullanarak bu uygulamayı da kendi içerisinde halleder.

ÖU1.4 Emek ve maliyet hesaplarının yapılması = XP mevcut kaynakları değerlendirerek zaman * eleman * adım hesabı ile emek ve maliyet hesapları yapılır.Planlama oyunu sayesinde emek ve maliyet hesaplarının yapılması kolaylaşır.

ÖU2.1 Bütçe ve zaman çizelgelerinin belirlenmesi = XP pratiği olan haftada 40 saat çalışma sayesinde elemanların çalışma süreleri hesap edilmeye çalışılır. Ayrıca her müşteri kartı için oluşturulan bir bütçe hesaplaması da vardır.

ÖU2.2 Proje risklerinin belirlenmesi = XP'ye göre projelerde risk vardır. Risklerden kaçınılması imkânsızdır. Risklerin varlığını bilerek ona göre hareket edilir. Önemli olan risklerden kaçmak yerine bir an önce risk ile karşılaşmaktır. Küçük sürümler ile yanlışlıklar erken safhalarda görülmesi sayesinde proje içerisinde oluşabilecek üretim risklerini azaltmaya çalışır.Kısa dönemli planlamalar ve değişikliğe açık bir yapısının olması risklerin azaltılmasını sağlar.

ÖU2.3 Veri yönetimi ve planlama = XP'de iletişim çok önemli bir yer tutar. İletişim müşteri ile yazılımcı arasındaki veri akışları ve yöntemleri belirlenir.Planlama oyunu sayesinde veri yönetimi işi gerçekleştirilir.

ÖU2.4 Proje kaynaklarının planlanması = XP'de insan kaynakları planlaması yoktur. Bu tarz yönetimsel işleri kendi yapısı içerisinde barındırmaz. Yazılım işlerine odaklıdır.

ÖU2.5 Gerekli yeteneklerin ve yazılım üretme bilgisinin sağlanması = XP yüksek kalitede eleman ister. Çiftli programlama sayesinde bilginin diğer programcıya geçmesi sağlanır.

ÖU2.6 Proje üreticilerinin katılımının planlanması = XP'de yazılımcılar bir arada aynı ortamda bulunurlar. Yazılımcıların birbirlerine yardım etmeleri esastır. Kod ortak mülktür anlayışı vardır.

ÖU2.7 Proje planının oluşturması = Proje planları müşteri kartları üzerindedir. XP kendi yapısı içerisinde proje başında mümkün olduğunca basit ama diğer taraftan genişlemeye imkân veren bir yapı tasarımı oluşmasını sağlayan bir plan gerçekleştirimi ister. Planlama oyunu bu işin gerçekleştirilmesi sağlar.

ÖU3.1 Projeden etkilenen diğer projelerin planlarına bakılması = XP'de her proje kendi yapısı içerisinde değerlendirilir. Bir projeden elde edilen bilgiler diğer projeler içerisinde direk olarak kullanılmaz.

ÖU3.2 İş ve kaynak yeterlilik seviyelerinin belirlenmesi = XP'de yeterlilik kısmı, teknik kartlarda istenilenlerin yapılıp yapılmadığı ile doğru orantılıdır. Teknik kartlarda yazılan işler gerçekleştirilip gerçekleştirilmediğe bakılır. Başta oluşturulan planlama oyununa göre kaynaklar belirlenir.

ÖU3.3 Proje taahhütlerinin oluşturulması = Proje taahhütleri müşteri kartlarına yazdıklarıdır. Bunlar koçlar tarafından onaylandıktan sonra taahhüt kapsamına girer.

6.1.3 Proje izleme ve takip:

Projenin durumunun tespiti ile ilgilidir. Bu sayede projenin ne ölçüde ilerlediği hakkında bilgi edinilebilir. Sürümlerin kısa sürelerde bir oluşturulması projeyi izlemeyi kolaylaştırır.Ayrıca müşteri kart sayıları sayesinde proje içerisinde nerede olduğumuz sorusunun da cevabı alınır.

XP proje takibi için ve projenin tamamını görebilmek için büyük görsel bir grafikten yararlanır.Bu grafik üzerinde yapılanlar ve yapılmayanlar işaretlenerek projenin hızı istenilenlerin ne derece karşıladığı gibi bazı bilgiler ile proje izleme ve takibi ile ilgili tespitler yapılabilir. Müşteriden gelen bilgiler oluşturulan planlama oyunu bu işi gerçekleştirmek için çok önemlidir.

ÖU1.1 Proje planlarının takibi = XP proje planlarının tahminlerinde teknik ve müşteri kartlarında belirlenen başlangıç ve bitiş tarihlerinden yararlanır. Bu tarihleri proje içerisindeki koç tarafından belirlenir.

ÖU1.2 Taahhütlerin takibi = Kartlar ile projede verilen taahhütlerin gerçekleşip gerçekleşmediğine bakılır.

ÖU1.3 Proje risklerinin takibi = XP pratikleri sayesinde riskler bertaraf edilmeye çalışılır.

ÖU1.4 Verilerin yönetimlerinin takibi = Veri yönetimleri müşteri kartları ile yapılır.

ÖU1.5 Proje Paydaşlarının katılımlarının takip edilmesi = Her kez aynı odada bütün kaynaklar tek bir yerde olduğundan proje katılımları rahatlıkla takip edilebiliyor.

ÖU1.6 Gelişme toplantılarının yapılması = XP müşteri yazılım ekibi içerisinde prensibi ve günlük ayaküstü yapılan toplantılar ile projenin gelişmesi izlenir.Ayrıca XP'de yapılan periyodik toplantılarda vardır.

ÖU1.7 Kilometre taşı toplantılarının yapılması = Periyodik yapılan toplantılar.

ÖU2.1 Sorunların analizi = Yönetime yazılan raporlar

ÖU2.2 Düzenleyici aksiyonların tespiti = Sürekli yapılan iterasyonlar, günlük ayakta yapılan toplantılar.

ÖU2.3 Düzenleyici aksiyonların yönetilmesi = Planlama oyunu sayesinde eksiklikleri ve yanlışlıkları düzeltmeye çalışır. Bu işle koç ve yazılım ekibi bizzat ilgilenir.

6.1.4 Tedarikçi sözleşme yönetimi:

Tedarikçi sözleşme yönetiminin amacı, tedarikçiden ürün satın işlemini taraflar arasında imzalanmış bir sözleşme ile yönetmektir. Temel faaliyetleri satın alma işlerinin belirlenmesi, sözleşmenin oluşturulması, güncelliğin satın alınan sözleşmede istenilenlerin yerine getirilmesi, kabul süresi gibi aşamaları vardır. XP satın alma, tedarikçi sözleşmesi oluşturma gibi alanlar ile ilgilenmez. XP çalışma alanının dışında bir konudur.

6.1.5 Süreç ve ürün kalite güvencesi:

Süreç ve ürün kalite güvencesinin amacı, çalışanlara ve yöneticilere, süreçler ve ilgili iş ürünleri ile ilgili, tarafsız gözlemler sunmaktır. Uygunsuzlukları tespit etmek, belgelemek, proje ekibini bilgilendirmek gibi faaliyetleri vardır. Her ne kadar SEPG gibi yazılım kalitesi ile ilgilenen ayrı bir ekip olmasa da ürünlerin ve süreçlerin güvencesi ve isterlerin karşılanması ile ilgilenen çift programlama tekniği sayesinde yazılımcılar birbirlerinin yaptıklarından haberdardırlar. Bu sayede oluşabilecek yazılımsal uygunsuzluklar çok daha kısa süreler içersinde bertaraf edilebilir. Doğrudan olmasa da dolaylı bir kalite güvence aktivitesi XP içerisinde vardır.

Ayrıca başta oluşturulan büyük görsel grafik sayesinde süreçlerin bir arada düzenli bir yapıda olması sağlanır. Ölçüm sorumlusu da süreç ve ürün kalitesinden sorumludur..

ÖU1.1: Süreçlerin tarafsızca değerlendirilmesi = Çiftli programlama tekniği ile bir yazılımcının yaptığından diğer yazılımcılar tarafından değerlendirilir. Ama değerlendiren kişi de o yazılımı yazan kişi olduğundan çok fazla objektif olması düşünülemez. Koçlarda süreçleri değerlendirme işleri ile uğraşırlar.

ÖU1.2: İş ürünleri ve hizmetlerin tarafsız olarak değerlendirilmesi = Bu iş içinde çift programcı desteği kullanılır. İlave olarak diğer ürünü oluşturan yazılımcıların yapacakları kod gözden geçirmeleri ve süreç uzmanı olan koçların yazılımları değerlendirmesi sayesinde bu aktivite geçekleştirilir.

ÖU2.1: Uygunsuzlukların ilan edilmesi ve çözünüldüğünden emin olunması = Uygunsuzlukların olmaması koçun proje içerisindeki ana görevidir. Bu sayede olabilecek aksaklıklar giderilmeye çalışılır. Kodun ortak olarak üretilmesi sayesinde uygunsuzluklar daha erken safhalarda belirlenir. Bu bulunan uygunsuzluklar aynı ortam içerisinde çalışan yazılımcılar tarafından sorumlu yazılımcıya iletilebilir.

Burada XP için önemli olan bir kriteri olan iletişimin büyük bir etkisi vardır. Günlük yapılan ayaküstü toplantılarda da uygunsuzlukların giderilip giderilemediği hakkında görevli yazılımcı bilgi verebilir.

ÖU2.2: Kayıtların oluşturulması = Kayıtlar müşteri kartları ve teknik kartlardadır. Ayrıca proje ile ilgili başlangıç bilgileri her kartın ne kadar sürede bittiği kalan kart sayısı bilgisi gibi bilgiler zaman çizelgeleri ve kontrol çizelgelerde kullanılarak proje hakkındaki ana bilgiler kısmında tutulur. Projenin temel bilgileri olan proje başlangıç tarihi, ister bilgileri bitiş tarihi gibi bilgileri içeren kayıtlar vardır.

6.1.6 Konfigürasyon yönetimi:

Konfigürasyon yönetiminin amacı, konfigürasyon tanımlama, konfigürasyon kontrol, konfigürasyon durum değerlendirme ve konfigürasyon denetleme faaliyetlerini kullanarak iş ürünlerinin tutarlılığını ve bütünlüğünü oluşturmak ve korumaktır. Temel sürümleri oluşturmak, konfigürasyon parçalarının belirlenmesi gibi faaliyetleri vardır.

XP tam olarak konfigürasyon yönetimi ile bire bir uyuşmasa da düzgün çalışan program, küçük sürümler, sürekli entegrasyon, refactoring, kodun ortak sahiplenmesi gibi pratikleri ile bu işleri gerçekleştirmeye çalışır.

ÖU1.1: Konfigürasyon parçalarının belirlenmesi = XP'de ortaya konulan, birkaç haftada bir ortaya çıkarılan sürümler, tekrardan bir önceki sürüme geri dönebilirlik gibi avantajları ile konfigürasyon yönetimi işi gerçekleştirilmeye çalışılır. Sistem içerisindeki parçaların bir araya getirilmesi sayesinde sürümlerin ortaya çıkartılması sağlanır. Tabi ki XP yapısı gereği sürekli değişen koşullara ayak uydurma zorunluluğundan ötürü konfigürasyon parçalarının belirlenmesi işi dikkatle ele alınmalıdır.

ÖU1.2: Konfigürasyon yönetim sisteminin oluşturulması = Konfigürasyon işlerinin yönetimi yazılım ekibinin başında bulunan koç tarafından karşılanır.

ÖU2.1:Değişiklik isteklerinin takibi = Proje içerisinde oluşan hikâyeler ile yapılır. Planlama oyunu ve bu oyun için gerekenler belirlenir.
ÖU2.2: Konfigürasyon parçalarının kontrol edilmesi = Sistem içerisinde mevcut bütün parçaların kontrol edilmesi ile sağlanır. Burada gelecekte ne gibi isterler gelebilir sorusu üzerinde düşünülür ve ona göre genişlemeye uygun ve basit bir sistem geliştirilmeye çalışılır.Sürekli entegrasyon yapılır.

ÖH3: Bütünlüğün sağlanması =XP yapısı içerisinde ortak kod ve çalışma alanının sağladığı avantajlar ile bütünlüğü sağlar. Bütünlüğün sağlanması için, bütünü

oluşturan parçalar belli zamanlarda bir araya getirilerek bütünlük oluşturulur.Entegrasyon vardır.

ÖU3.1: Konfigürasyon yönetim kayıtlarının oluşturulması = XP'de bu tarz bir kayıt sistemi yoktur.

ÖU3.2: Konfigürasyon denetimlerinin oluşturulması = Denetimler koçlar tarafından yapılır. Belli zamanlarda oluşturulması gereken iterasyonlar da koçlar tarafından yapılır. İlave olarak çiftli programcı desteği de konfigürasyon denetimlerinin oluşmasını sağlar.

6.1.7 Ölçme ve analiz :

Ölçme ve analizin amacı, yönetimin bilgi ihtiyacına destek olmak için uygun ölçme yeteneklerini geliştirmek ve devamlılığını sağlamaktır. Ölçme hedefi belirlenir, ölçütler saptanır, ölçme veriler analizi yapılması gibi aktiviteleri vardır. XP bu aktivitelere günlük yapılan toplantılar, kısa sürede çıkartılan sürümlerdeki isterlerin karşılama oranları, mevcut kartların sayısını başlangıçtakine oranı gibi bazı temel ölçüm ve analiz bilgileri ile karşılık verir. XP ölçme ve analiz aşamasına yaklaşımı CMMI 2 seviye süreçlerinden olan ölçme ve analiz istenilenleri ile tam olarak örtüşmese de bazı uygulamları tam olarak yerine getirmektedir.

ÖU1.1 Ölçme hedefinin belirlenmesi = XP'de ana ölçme hedefi mevcut projenin bitiş tarihidir. Bu tarih içerisinde müşteri isterlerinin olduğu kartların ne kadarının karşılandığıdır. İsterlerin belirlenmesidir. Yönetici, koç ve müşteri ile bu tarih hesaplaması yapılır.

ÖU1.2: Ölçütlerin saptanması = Ölçme hedefleri mevcut kart sayısı, proje bitiş tarihi proje süresi gibi ana ölçütlerdir.

ÖU1.3: Veri toplama ve saklama prosedürlerinin belirlenmesi = Ölçme ve veri toplamanın yapılması işi A4 kağıdının yarısı büküklüğünde olan kartlara yazılarak yapılır. Bütün veriler bu biçimi belirlenmiş kartlar üzerindedir.

ÖU1.4: Analiz prosedürlerinin belirlenmesi = Prosedürlerin gerçekleştirilip gerçekleştirilemeyeceğine ve ne kadar sürede gerçekleşeceğine koçlar karar verir. Daha sonra bu prosedür teknik kartlar üzerine parçalanarak yazılır.

ÖU2.1: Ölçme verilerinin toplanması = Bu işi XP proje ekibinin içinde olan ölçüm sorumlusu yapar. Ölçüm sorumlusu; gerçekleşen değerleri bir araya getirir. Gerçekleşen değerler ile önceden tespit edilene değerleri karşılaştırılması yapılır. Gidişatın tespit edilmesini, ölçünlerin doğru alınması işlerinden sorumludur.

ÖU2.2: Ölçme verilerinin analiz edilmesi = Ölçüm sorumlusu tarafından yapılacak bir başka iştir. Bu sayede projenin durum tespiti kolaylaşır.

ÖU2.3: Veri ve analiz sonuçlarının saklanması = Analiz sonuçları yönetici ve koçlar tarafından muhafaza edilir.

ÖU2.4: Sonuçların ilan edilmesi = Sonuçlar yönetici tarafından her iterasyonda bir ilan edilir.

6.2 CMMI ve XP Pratiklerinin Uyum Oranları

CMMI 2. seviyesindeki 7 süreç alanına baktığımızda pek çok proje alanlarının, XP proje yaşam çevrimi içerisindeki pratikleri kullanarak karşılandığını görüyoruz. Burada baktığımızda çok büyük olmayan şirketlerin CMMI 2.seviye olmak için XP pratiklerini kullanabilecekleri sonucuna ulaşıyoruz.

7. Sonuçlar

Bu tez ile birlikte CMMI değerlendirmesi için XP'nin nasıl kullanılacağı ve CMMI'ın isterlerini XP'in nasıl sağlayacağı sorusuna cevap aramaktadır. Özellikle Türkiye gibi yazılım üreten şirketlerin küçük ve orta büyüklükte olduğu bir ülkede CMMI ile XP kullanımının bir arada yapılmasının ne gibi avantajlar sağlayacağı da aşağıda maddeler halinde anlatılmıştır.

- XP kullanımı ile, yazılım üretim hızı artırılabilir ve bunun sonucu olarak müşteri memnuniyeti de artacaktır.

- XP kullanılarak CMMI 3.seviye isterleri, ek uygulamalarla sağlanılabilir.

- Şirketin iç yapısı, üretilen yazılımların biçimi ve müşteri profili uygunsa XP alınacak eğitimler doğrultusunda kullanılabilir.

- XP kullanımı ile maalesef CMMI 4.seviye ve CMMI 5.seviye isterleri karşılamamaktadır.

Yukarıdaki ve 6.bölümde belirtilen hususlara uygulandığında özellikle küçük ve orta ölçekli şirketlerin, XP uygulayarak CMMI 2.düzeyi isterleri karşılanabilir.

KAYNAKLAR

- [1] Dr.Erhan Sarıdoğan 2004 "Yazılım Mühendisliği" Papatya Yayıncılık s 111
- [2] Mustafa Alkan 2004 "Yazılım Mühendisliği Nedir?" http://www.csharpnedir.com/makalegoster.asp?MId=230
- [3] Oktay Altunergil 2003 "Yazılım Ciddi ve Zor Bir İştir" http://www.belgeler.org/howto/ozguryazilim.html
- [4] Dr.Erhan Sarıdoğan 2004 "Yazılım Mühendisliği" Papatya Yayıncılık s 109
- [5] Mustafa Alkan 2004 "YazılımSıfınları" http://www.csharpnedir.com/makalegoster.asp?MId=230
- [6] Mustafa Alkan 2004 "YazılımSıfınları" http://www.csharpnedir.com/makalegoster.asp?MId=230 http://www.csharpnedir.com/makalegoster.asp?MId=230
- [7] Özgür Yoğurtcu 2005 "Yazılım Projeleri Neden başarısız" http://www.veripark.com/newsletter/mart06/mart01.html?7
- [8] Eser Sevinç 2005 "Bilgisayar Tarihi" http://www.ymm.net/e-ticaret/bilgisayartarihi.htm
- [9] Dr.Erhan Sarıdoğan 2004 "Yazılım Mühendisliği" Papatya Yayıncılık s 112
- [10] Mustafa Alkan 2004 "Yazılım Mühendisliği Nedir?" http://www.csharpnedir.com/makalegoster.asp?MId=230
- [11] Dr. Erhan Sarıdoğan 2004 "Yazılım Mühendisliği" Papatya Yayıncılık s 23
- [12] Meriç Merih Aykol 2003 "Yazılım Kalitesi Bilinci" s33
- [13] Dr.Erhan Sarıdoğan 2004 "Yazılım Mühendisliği" Papatya Yayıncılık s 113
- [14] Turgay Aytaç 2002 "Yazılım İnsan ve Kalite Etkileşimi"
- [15] Turgay Aytaç 2002 "Kaliteli Yazılım Üretimi" http://www.btnet.com.tr/koseyazi.phtml?kategori_id=14&yazi_id=190000098
- [16] Dr. Erhan Sarıdoğan 2004 "Yazılım Mühendisliği" Papatya Yayıncılık s 79
- [17] Dr.Erhan Sarıdoğan 2004 "Yazılım Mühendisliği" Papatya Yayıncılık s 119
- [18] Dr.Erhan Sarıdoğan 2004 "Yazılım Mühendisliği" Papatya Yayıncılık s 114
- [19] Dr.Erhan Sarıdoğan 2004 "Yazılım Mühendisliği" Papatya Yayıncılık s 124
- [20] Orhan Kalaycı 2005 "CMMI Eğitimi" s115 www.nitelik.net
- [21] Delphi Ekibi 2005 "RAD NEDİR" http://www.delphiturkiye.com/dgiris.htm

- [22] İGEME 2004 “Kalite Nedir” http://www.igeme.org.tr/TUR/pratik/kalite.pdf
- [23] SAMİ ÖZTÜRK 2004 **“**ISO 9000 Toplam Kalite Güvence Sistemleri**”** s34
- [24] Türker BAŞ 2003 “ISO 9000:2000 Kalite Yönetim Sistemi” s35
- [25] Türker BAŞ 2003 “ISO 9000:2000 Kalite Yönetim Sistemi” s35
- [24] Türker BAŞ 2003 “ ISO 9000 TOPLAM KALİTE GÜVENCE SİSTEMLERİ“ http://www.kadem.com.tr/turkce/urun_detay.aspx?id=24
- [26] SGS 2003 “ISO 9001:2000“ http://www.tr.sgs.com/tr/iso_9001_2000.htm?serviceId=10954&lobId=15954
- [27] TSE 2004 “ISO Nedir” http://www.tse.org.tr/Turkish/KaliteYonetimi/9000bilgi.asp
- [28] BSİ Management System “KYS Belgelendirmesi “ http://www.bsi-turkey.com/Kalite/QMStescil/
- [29] Mehmet Özkan 2002 “ISO 9001’e Giriş” http://www.bilgiyonetimi.org/cm/pages/mkl_gos.php?nt=43
- [30] “Yazılım İçin Ömür Devri Boyunca Birleştirilmiş NATO Kalite Gereksinimleri AQAP-160 BASKI 1” 2001 http://www.msb.gov.tr/Birimler/KaliteYonD/pdf/AQAP_160.pdf s 5
- [31] “Yazılım İçin Ömür Devri Boyunca Birleştirilmiş NATO Kalite Gereksinimleri AQAP-160 BASKI 1” 2001 http://www.msb.gov.tr/Birimler/KaliteYonD/pdf/AQAP_160.pdf s 8
- [32] “Yazılım İçin Ömür Devri Boyunca Birleştirilmiş NATO Kalite Gereksinimleri AQAP-160 BASKI 1” 2001 http://www.msb.gov.tr/Birimler/KaliteYonD/pdf/AQAP_160.pdf s 16
- [33] “Yazılım İçin Ömür Devri Boyunca Birleştirilmiş NATO Kalite Gereksinimleri AQAP-160 BASKI 1” 2001 http://www.msb.gov.tr/Birimler/KaliteYonD/pdf/AQAP_160.pdf s 24
- [34] SWEBOK 2004 “Guide to the Software Engineering Body of Knowledge” www.swebok.org/swebok_guide_2005.pdf
- [35] Eric Williams ,1998 “Software Life Cycle Processes”
- [36] Dr.M.Erhan Sarıdoğan 2004 “Yazılım Mühendisliği” s 404
- [37] Software Engineering Institute 2002 “CMM”
- [38] Juan Hjenling 2004 “CMMI İntigration” A course Paper s 5
- [39] Mary Baht Chriessis,Mikeconrad,Sandy Scrum “Guidelines for Process Integration and Product Improvement ” Addison Wesley s 27
- [40] Bilgi Yönetimi 2005 “Yazılım Tarihi” www.bilgiyonetimi.org\yazilim_tarihi.htm
- [41] Carnegie Mellon University Software Engineering Institue 2001 “History of CMMI ” S 77
- [42] Carnegie Mellon University 2002 “History of CMMI ” Software Engineering Institue S 17

- [43] Carnegie Mellon University Software Engineering Institue 2001 “History of CMMI ” S 71
- [44] BT NET “Yüksek Seviyedeki Firmaların Uygulamaları” www.btnet.com\ysfu.html
- [45] Orhan Kalaycı 2005 “CMMI Eğitimi” www.nitelik.net S 117
- [46] Carnegie Mellon UniversitySoftware Engineering Institue 2001 “History of CMMI ” S 10
- [47] Thomas Gibbon, 1999 “A Business Case for SPI REVİSED” DACS s 26
- [48] Thomas Gibbon,1999 A Business Case for Software Process Improvement Revised DoD Data & Analysis Center for Software (DACS),Sayfa 5

- [49] Orhan Kalaycı 2005 “CMMI Eğitimi” www.nitelik.net S 139
- [50] Thomas Gibbon ,1999 A Business Case for Software Process Improvement Revised DoD Data Analysis Center for Software (DACS),S 26”
- [51] Thomas Gibbon ,1999 A Business Case for Software Process Improvement Revised DoD Data Analysis Center for Software (DACS),S 30”
- [52] Mary Baht Chriessis,Mikeconrad,Sandy Scrum “Guidelines for Process Integration and Product Improvement ” Addison Wesley s 590
- [53] Mary Baht Chriessis,Mikeconrad,Sandy Scrum “Guidelines for Process Integration and Product Improvement ” Addison Wesley s 759
- [54] Mary Baht Chriessis,Mikeconrad,Sandy Scrum “Guidelines for Process Integration and Product Improvement ” Addison Wesley s 687
- [55] Mary Baht Chriessis,Mikeconrad,Sandy Scrum “Guidelines for Process Integration and Product Improvement ” Addison Wesley s 786
- [56] Mary Baht Chriessis,Mikeconrad,Sandy Scrum “Guidelines for Process Integration and Product Improvement ” Addison Wesley s 489
- [57] Mary Baht Chriessis,Mikeconrad,Sandy Scrum “Guidelines for Process Integration and Product Improvement ” Addison Wesley s 305
- [58] Mary Baht Chriessis,Mikeconrad,Sandy Scrum “Guidelines for Process Integration and Product Improvement ” Addison Wesley s 420
- [59] Mary Baht Chriessis,Mikeconrad,Sandy Scrum “Guidelines for Process Integration and Product Improvement ” Addison Wesley s 467
- [60] Mary Baht Chriessis,Mikeconrad,Sandy Scrum “Guidelines for Process Integration and Product Improvement ” Addison Wesley s520
- [61] Mary Baht Chriessis,Mikeconrad,Sandy Scrum “Guidelines for Process Integration and Product Improvement ” Addison Wesley s 731
- [62] Mary Baht Chriessis,Mikeconrad,Sandy Scrum “Guidelines for Process Integration and Product Improvement ” Addison Wesley s 470
- [63] Mary Baht Chriessis,Mikeconrad,Sandy Scrum “Guidelines for Process Integration and Product Improvement ” Addison Wesley s 643
- [64] Mary Baht Chriessis,Mikeconrad,Sandy Scrum “Guidelines for Process Integration and Product Improvement ” Addison Wesley s 161

- [65] Orhan Kalayci 2005 “CMMI Eğitimi” www.nitelik.net s 198
- [66] Carnegie Mellon University 2005 “Process Maturity Profile CMMI Scampi” Software Engineering Institute
- [67] Thomas Gibbon,1999 A Business Case for Software Process Improvement Revised DoD Data & Analysis Center for Software (DACS),Sayfa 6
- [68] Karl E. Wiegers 1996 “Creating a software engineering culture” Dorset House Publishing s27
- [69] Gartner Raporları 2002
- [70] Mary Baht Chriessis,Mikeconrad,Sandy Scrum “Guidelines for Process Integration and Product Improvement ” Addison Wesley s 27
- [71] Orhan Kalaycı 2005 “CMMI Eğitimi” s 227
- [72] BT NET 2002 “Yüksek Olgunluga Sahip Firmaların Uygulamaları” www.btnet.com\yosfu.htm
- [73] Kent Beck 2002 “eXtreme Programming” Addison-Wesley s 9
- [74] John Brewer 2004 “Extreme Programming FAQ” www.java.com\FAQ.HTM
- [75] Orhn Kalaycı 2005 “CMMI Eğitimi” s 201
- [76] Shivraj Kanungo and Asha Goyal 2004 “CMMI Implementation” *s12*
- [77] Kent Beck 2002 “eXtreme Programming” Addison-Wesley s 21
- [78] Kent Beck 2002 “eXtreme Programming” Addison-Wesley s 63
- [79] Kent Beck 2002 “eXtreme Programming” Addison-Wesley s 15
- [80] Kent Beck 2002 “eXtreme Programming” Addison-Wesley s 27
- [81] Agile Team 2005 “12 Practice” www.agilemanifesto.com\12Practice.html
- [82] XPorganization 2005 “XP ?” www.xporganization.com\xp.html
- [83] Julio Ariel Hurtado 2003 Agile Methods s 2
- [84] Kent Beck 2002 “eXtreme Programming” Addison-Wesley s 85
- [85] Kent Beck 2002 “eXtreme Programming” Addison-Wesley s 117
- [86] Çağatay Çalal 2005 “RUP&XP Karşılaştırılması” C# yazılımMerkezi www.csharpnedir.com/makalegoster.asp?mid=459
- [87] Orhan Kalaycı 2006 “XP Eğitimi” www.nitelik.com s 155
- [88] Orhan Kalaycı 2006 “XP Eğitimi” www.nitelik.com s 162
- [89] Orhan Kalaycı 2006 “XP Eğitimi” www.nitelik.com s 232

ÖZGEÇMİŞ

1980 yılında Ankara'da doğdum. Lise eğitimini Amasya Lisesi'nde tamamladım. 1999 – 2003 yılları arasında T.C. Maltepe Üniversitesi Bilgisayar Mühendisliği Bölümü'nde öğrenim gördüm. 2003 yılında mezun olduktan sonra T.C. Maltepe Üniversitesi'nde Uzman olarak çalışmaya başladım. Halen aynı görevi sürdürmekteyim.