ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ

MÜHENDİSLİK- MİMARLIK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

2025-2026 GÜZ YARIYILI

YAZILIM MÜHENDİSLİĞİ DERSİ

Doç. Dr. Nihat ADAR

Dr. Öğr. Üyesi Uğur GÜREL

ISO/IEC 12207:1995, ISO/IEC 15026, Yazılım Yaşam Döngüsü Süreçleri, Yazılım Geliştirme Standartları

Grup No:1

Grup Üyeleri:

Şevval Ayça Çerence - 152120211128

Nisanur Pehlivan - 152120221054

Merve Selçuk - 152120211116

Pelin Özer – 152120231100

(Tüm üyeler katkı sağlamıştır ve katkı oranları eşittir.)

İçindekiler

Abstract

Bu rapor, yazılım yaşam döngüsünün yönetimi için temel çerçevelerden biri olan ISO/IEC 12207:1995 standardını; süreç sınıfları, roller, süreçlerin amaçları ve yaşam döngüsü boyunca üretilmesi beklenen tipik çıktılar üzerinden ayrıntılı biçimde ele almaktadır. Ayrıca kritik özelliklerin (örneğin güvenlik, emniyet ve güvenilirlik) kanıta dayalı şekilde gerekçelendirilmesini hedefleyen ISO/IEC 15026 standart ailesi incelenmekte; özellikle güvence dosyası (assurance case) yaklaşımı (ISO/IEC 15026-2), integrity level (integrity level) kavramı (ISO/IEC 15026-3) ve güvence faaliyetlerinin yazılım yaşam döngüsü süreçlerine nasıl entegre edildiği (ISO/IEC/IEEE 15026-4) sistematik biçimde açıklanmaktadır. ISO kataloğunda ISO/IEC 12207:1995 standardının withdrawn statüsünde olduğu ve daha güncel sürümlerinin bulunduğu belirtilmekle birlikte, 1995 sürümü yazılım yaşam döngüsü süreç mantığını öğrenmek açısından güçlü bir temel referans niteliğini korumaktadır.

1. Giriş

Yazılım sistemlerinin kapsamı ve karmaşıklığı arttıkça, proje çıktılarının yalnızca "çalışır bir ürün" üretmekle sınırlı kalmadığı; aynı zamanda kalite, izlenebilirlik, denetlenebilirlik ve sürdürülebilir bakım gibi boyutlarıyla da güvence altına alınması gerektiği açıkça ortaya çıkmaktadır. Güncel mühendislik uygulamalarında yazılım; çok sayıda paydaşın beklentisini aynı anda karşılamak zorunda olan, farklı teknoloji bileşenleriyle sürekli etkileşen ve çoğu zaman değişen gereksinimler altında evrilen bir üründür. Bu nedenle yazılım geliştirme faaliyetlerinin kurumsal ölçekte yönetilebilmesi için süreçlerin açık biçimde tanımlanması, rollerin netleştirilmesi, iş ürünlerinin standartlaştırılması ve değişikliklerin kontrollü biçimde yönetilmesi kritik önem taşır.

Bu bağlamda ISO/IEC 12207:1995, yazılım yaşam döngüsü boyunca yürütülen faaliyetleri "süreç" mantığında ele alarak; edinimden bakıma kadar uzanan bir çerçeve sunar. Standart; yalnızca geliştirme adımlarını değil, kalite güvencesi, konfigürasyon yönetimi, doğrulama, geçerleme, denetim ve problem çözümü gibi destekleyici mekanizmaları da yaşam döngüsünün ayrılmaz bir parçası olarak konumlandırır. Böylece proje yönetimi ve mühendislik faaliyetleri, belirli bir disiplin içinde yürütülerek tekrarlanabilirlik ve ölçülebilirlik kazanır.

Öte yandan, özellikle güvenlik, emniyet, bütünlük ve süreklilik gibi kritik niteliklerin önemli olduğu sistemlerde, süreçlerin tanımlanması tek başına yeterli olmayabilmektedir. Bu tür sistemlerde "ürünün ve sürecin uygun yürütüldüğü" iddiasının, paydaşları ikna edecek şekilde kanıta dayalı olarak gerekçelendirilmesi beklenir. ISO/IEC 15026 standart ailesi bu ihtiyaca yanıt verir; seçilmiş bir iddia için argümantasyonun nasıl kurulacağını, hangi tür kanıtların nasıl yapılandırılacağını ve integrity level gibi yaklaşımlarla güvence yoğunluğunun nasıl belirleneceğini ele alır. Bu perspektif, yalnızca kaliteyi ölçmeyi değil; aynı zamanda kaliteye ilişkin iddiaları savunulabilir ve denetlenebilir bir çerçevede sunmayı amaçlar.

Dolayısıyla yazılım mühendisliği uygulamalarında 12207'nin süreç temelli disiplini ile 15026'nın kanıt temelli güvence yaklaşımı birlikte değerlendirildiğinde, yaşam döngüsü yönetimi "ne yapılacak?" sorusuyla sınırlı kalmaz; aynı zamanda "neden güvenilebilir?" sorusunu da sistematik biçimde yanıtlayabilen bir yapıya kavuşur. Bu rapor, söz konusu iki yaklaşımı bütüncül biçimde ele alarak, yazılım geliştirme süreçlerinde birlikte konumlandırılmalarına yönelik kavramsal ve uygulama odaklı bir çerçeve oluşturmayı hedeflemektedir.

1.1 Amaç

Bu çalışmanın temel amacı; ISO/IEC 12207:1995 standardının yazılım yaşam döngüsü süreçlerini süreç–rol–çıktı ekseninde açıklamak ve ISO/IEC 15026 standart ailesinin assurance case, integrity level ve yaşam döngüsüne entegrasyon yaklaşımlarını ele alarak, bu iki standardın yazılım geliştirme pratiklerinde nasıl birlikte konumlandırılabileceğini ortaya koymaktır.

1.1.1 Alt Amaçlar

Bu temel amaç doğrultusunda çalışma aşağıdaki alt hedefleri gerçekleştirmeyi amaçlar:

1. **Süreç Mimarisinin Açıklanması:**
   ISO/IEC 12207 kapsamında yer alan **birincil** (edinim, tedarik, geliştirme, işletim, bakım), **destek** (dokümantasyon, konfigürasyon yönetimi, kalite güvencesi, doğrulama, geçerleme, gözden geçirme, denetim, problem çözümü) ve **kurumsal** (yönetim, altyapı, iyileştirme, eğitim) süreçlerin kapsamını; süreçler arası ilişkiyi ve organizasyon içindeki rol dağılımını sistematik biçimde açıklamak.

2. **İş Ürünleri ve İzlenebilirliğin Yapılandırılması:**
   Süreçlerden üretilen iş ürünlerini (gereksinim dokümanları, mimari karar kayıtları, test raporları, değişiklik kayıtları vb.) sınıflandırmak; bu iş ürünlerinin izlenebilirlik zincirindeki yerini belirlemek ve yaşam döngüsü boyunca sürdürülebilir biçimde yönetilebilmesi için gerekli kontrol noktalarını tanımlamak.

3. **Güvence Mantığının Kavramsallaştırılması:**
   ISO/IEC 15026 çerçevesinde güvence yaklaşımını; "iddia–argüman–kanıt" ilişkisi üzerinden açıklamak, güvence dosyası yapısının bileşenlerini ve güvence iletişimindeki rolünü ortaya koymak.

4. **Bütünlük Seviyesi (Integrity Level) Perspektifinin Uygulanması:**
   Kritik özellikler ve kritik bileşenler için güvence yoğunluğunun nasıl belirleneceğini; integrity level yaklaşımı üzerinden tartışmak ve bütünlük seviyesi arttıkça beklenen doğrulama-inceleme-kanıt gereksinimlerindeki değişimi açıklamak.

5. **Entegrasyon Çerçevesinin Kurulması:**
   12207 süreçlerinden üretilen iş ürünlerinin 15026 assurance case içinde "kanıt" olarak nasıl konumlandırılabileceğini göstermek; süreç planlaması ile kanıt üretiminin aynı yönetim çerçevesi içinde nasıl ele alınabileceğine ilişkin uygulanabilir bir model sunmak.

6. **Uygulama Rehberi ve Şablonların Sunulması:**
   Kurumsal kullanım için izlenebilirlik matrisi, değişiklik talebi, risk kaydı, gözden geçirme kontrol listeleri ve assurance case metin şablonları gibi yapı taşlarını; rapor içeriğiyle tutarlı biçimde örneklendirmek.

1.1.2 Çalışmanın Temel Sorunsalı ve Beklenen Sonuç

Bu çalışma, yazılım yaşam döngüsünde süreçlerin tanımlanmasının tek başına yeterli olmadığı; süreç çıktılarının aynı zamanda kanıt değeri taşıyacak biçimde yapılandırılmasının gerekli olduğu varsayımından hareket eder. Bu kapsamda beklenen sonuç; yazılım geliştirme faaliyetlerinin (i) süreç temelli yönetimi ve (ii) kanıt temelli güvence yaklaşımı ile birlikte ele alındığında, kalite ve güvenilirliğin daha şeffaf, izlenebilir ve denetlenebilir şekilde temellendirilebileceğini göstermektir.

1.2 Kapsam

Bu rapor; **ISO/IEC 12207:1995** standardının yazılım yaşam döngüsü süreçlerini tanımlayan yaklaşımını ve **ISO/IEC/IEEE 15026** standart ailesinin güvence (assurance) kavramını yapılandıran bileşenlerini, kavramsal çerçeve–süreç mimarisi–iş ürünleri–kanıt üretimi ekseninde incelemekle sınırlıdır. Bu kapsam; süreçlerin tanımlanması, süreçler arası etkileşimlerin açıklanması, rol ve sorumlulukların çerçevelenmesi, tipik çıktıların sınıflandırılması ve 15026 yaklaşımıyla bu çıktılardan güvence kanıtı türetilmesi gibi başlıkları içerir.

Yazılım projelerinin yalnızca "kod geliştirme" etkinliği olarak görülmesi; sözleşme ve tedarik yönetimi, gereksinimlerin belirlenmesi ve değişiklik kontrolü, tasarım kararlarının gerekçelendirilmesi, doğrulama ve geçerleme kanıtlarının üretilmesi, sürümleme, canlı işletim, bakım ve sürekli iyileştirme gibi faaliyetlerin göz ardı edilmesine yol açabilmektedir. Bu durum, özellikle rol tanımlarının belirsiz olduğu; dokümantasyon, konfigürasyon yönetimi ve kalite güvence mekanizmalarının etkin işletilmediği projelerde, hata yoğunluğunu ve teslimat riskini artıran yapısal zafiyetler doğurur. Bu raporda ele alınan standartlar, söz konusu zafiyetlerin farklı boyutlarına yanıt veren iki tamamlayıcı yaklaşımı temsil etmektedir: (i) yaşam döngüsü süreçlerinin ortak bir terminoloji ve disiplin altında tanımlanması (12207) ve (ii) seçilmiş bir iddianın kanıt temelli biçimde gerekçelendirilmesi (15026).

1.2.1 ISO/IEC 12207:1995 Kapsamı: Süreç–Rol–Çıktı Perspektifi

ISO/IEC 12207:1995, yazılım yaşam döngüsü boyunca yürütülecek faaliyetleri; süreçler, aktiviteler ve görevler şeklinde yapılandırarak edinim, tedarik, geliştirme, işletim ve bakım bağlamlarında uygulanabilir bir çerçeve sunar. Bu raporda 12207 yaklaşımı, "süreç–rol–çıktı" düzleminde ele alınmıştır. Buna göre:

- **Süreç düzeyi**: Birincil süreçler (edinim, tedarik, geliştirme, işletim, bakım), destek süreçleri (dokümantasyon, konfigürasyon yönetimi, kalite güvencesi, doğrulama, geçerleme, gözden geçirme, denetim, problem çözümü) ve kurumsal süreçler (yönetim, altyapı, iyileştirme, eğitim) arasındaki ilişki açıklanır.

- **Rol düzeyi**: Süreçlerin tipik paydaşlara/rollerine dağılımı; sorumluluk, hesap verebilirlik ve etkileşim noktaları üzerinden değerlendirilir.

- **Çıktı düzeyi**: Süreçlerin ürettiği iş ürünleri (gereksinim dokümanları, mimari/tasarım dokümanları, test planları ve raporları, değişiklik kayıtları, sürüm notları, denetim raporları vb.) sınıflandırılır; bu ürünlerin izlenebilirlik ve denetlenebilirlik açısından taşıdığı değer tartışılır.

Bu çerçeve, raporun ilerleyen bölümlerinde 15026 güvence yaklaşımının dayandığı "kanıt üretimi" mantığına altyapı oluşturur.

1.2.2 ISO/IEC/IEEE 15026 Kapsamı: Güvence Dosyası (Assurance Case) ve Bütünlük Seviyeleri (Integrity Levels)

ISO/IEC/IEEE 15026 ailesi, özellikle kritik özelliklerin önemli olduğu sistemlerde, bir iddianın nasıl gerekçelendirileceğine ilişkin sistematik bir güvence yaklaşımı sunar. Bu raporda 15026 ailesi; üç temel eksen üzerinden ele alınmıştır:

- **Güvence Dosyası (Assurance Case) (ISO/IEC/IEEE 15026-2):**
  Assurance case, üst seviye iddiaların alt iddialara ayrıştırılması; bu iddiaları destekleyen mantıksal argümanların kurulması; kanıt setinin tanımlanması ve varsayımların açık biçimde belirtilmesi yoluyla "neden güvenilebilir?" sorusuna yapılandırılmış yanıt üretmeyi hedefler. ISO/IEC/IEEE 15026-2:2022 standardının özetinde, assurance case'lerin yapı ve terminolojisine ilişkin gereksinimlerin tanımlandığı ve standardın assurance case geliştirme ve sürdürme için uygulanabilir olduğu belirtilmektedir.

- **Integrity Levels (Integrity level)(ISO/IEC 15026-3):**
  Integrity level yaklaşımı; sistem/ürün öğelerinin kritiklik düzeyine göre güvence yoğunluğunun belirlenmesini ve bu seviyelerin sağlandığının gösterilebilmesi için karşılanması gereken gereksinimleri ele alır. ISO/IEC 15026-3:2011 özetinde, integrity level kavramının tanımlandığı; integrity level'in sistemlere, yazılım

ürünlerine, öğelere ve ilgili dış bağımlılıklara atanmasına yönelik gereksinim ve yöntem önerilerinin yer aldığı ifade edilmektedir.

- **Yaşam Döngüsünde Güvence (ISO/IEC/IEEE 15026-4):**
  ISO/IEC/IEEE 15026-4:2021 özetinde, seçilmiş bir iddiaya ilişkin güvence sağlamak üzere iddianın gerçekleştirilmesi ve gerçekleştirildiğinin gösterilmesine yönelik rehberlik sunulduğu; bu rehberliğin ISO/IEC/IEEE 15288 ve ISO/IEC/IEEE 12207 üzerine inşa edilen süreç görünümü üzerinden verildiği belirtilmektedir.

2. Temel Kavramlar, Tanımlar ve Kısaltmalar

2.1 Temel Tanımlar

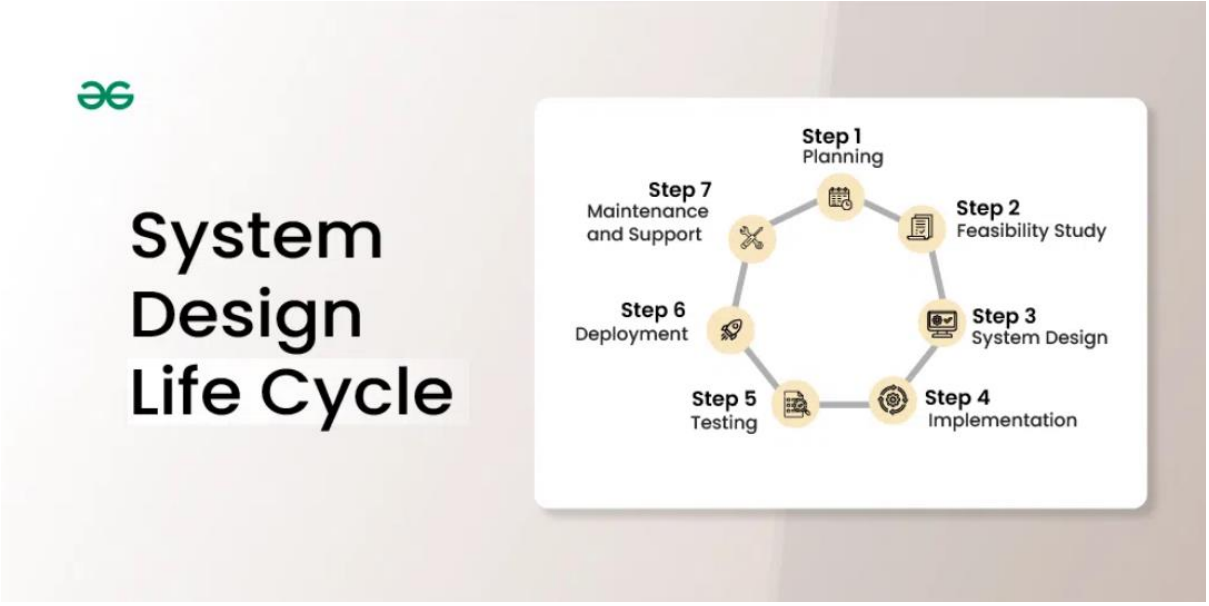
- **Yaşam döngüsü (life cycle):** Bir yazılım ürününün fikrin ortaya çıkışından edinim/tedarik kararlarına, geliştirmeden canlı işletime, bakımdan kullanımdan kaldırmaya kadar uzanan bütün aşamalar.
- **Süreç (process):** Belirli bir amacı gerçekleştirmek için tanımlanmış, girdileri çıktılara dönüştüren, yönetilebilir faaliyetler bütünü.
- **Aktivite (activity):** Bir sürecin içinde yer alan, daha dar kapsamlı iş grubu.
- **Görev (task):** Aktivitenin uygulanabilir en küçük iş adımı.
- **Çıktı / İş ürünü (work product):** Süreçler sonucunda üretilen teslim edilebilir veya denetlenebilir somut sonuç (doküman, kod, test raporu, kayıt, plan vb.).
- **Paydaş (stakeholder):** Üründen etkilenen veya ürünü etkileyen kişi/kurum (müşteri, kullanıcı, geliştirici, işletmeci, denetçi vb.).
- **Doğrulama (verification):** Gereksinim/tasarım/spesifikasyona uygunluk kontrolü. (“Ürünü doğru mu inşa ettik?”)
- **Geçerleme (validation):** Gerçek ihtiyacı/amaçlanan kullanımı karşılama kontrolü. (“Doğru ürünü mü inşa ettik?”)
- **Konfigürasyon öğesi (configuration item, CI):** Konfigürasyon yönetimi altında izlenen her bir bileşen (kaynak kod, gereksinim dokümanı, test senaryosu, derleme çıktısı vb.).
- **Baseline (temel çizgi):** Onaylanmış ve değişiklik kontrolüne alınmış sürüm seti (ör. “Gereksinim Raporu v1.0”).

2.2 Kısaltmalar

- **CM:** Configuration Management (Konfigürasyon Yönetimi)
- **QA:** Quality Assurance (Kalite Güvencesi)
- **V&V:** Verification & Validation (Doğrulama ve Geçerleme)

- **SIL/IL:** Safety/Integrity Level (Bütünlük seviyesi kavramı; 15026-3 özelinde "integrity level")

3. ISO/IEC 12207:1995 — Yazılım Yaşam Döngüsü Süreçleri



Şekil 3.1: Yazılım Yaşam Döngüsü Süreçleri

3.1 Amaç, Kapsam ve Uygulanabilirlik

ISO/IEC 12207:1995, yazılım yaşam döngüsü süreçleri için ortak bir terminoloji ve referans çerçevesi sunmayı amaçlayan; yazılımın ediniminden bakımına uzanan çalışmaların "hangi süreçler altında, hangi tür faaliyet ve çıktılarla" ele alınacağını tanımlayan bir standarttır.

Standardın kapsamı, yazılımın farklı bağlamlarda geliştirilmesini ve işletilmesini dikkate alacak şekilde geniş tutulmuştur. Bu çerçevede 12207; (i) yazılım içeren sistem edinimi (yazılımın daha büyük bir sistemin parçası olduğu durumlar), (ii) bağımsız yazılım ürünü geliştirme ve (iii) yazılım hizmetleri (geliştirme/işletim/bakım gibi hizmet odaklı sunum) bağlamlarında uygulanabilecek süreç kümelerini tanımlar. Standart, yazılımın gömülü olduğu senaryoları da kapsayacak biçimde yorumlanabildiği için firmware bileşenleri içeren sistemlerde de yaşam döngüsü faaliyetlerinin süreç disiplini altında ele alınmasına imkân verir.

Bu çerçeve, uygulamada farklı geliştirme ve tedarik modellerine uyarlanabilir. Örneğin kurumun yazılımı kendi bünyesinde geliştirdiği iç geliştirme senaryolarında edinen–tedarikçi ilişkisi aynı organizasyon içinde farklı birimler üzerinden kurgulanabilir. Dış tedarik senaryolarında ise süreçlerin bir kısmı tedarikçi tarafından yürütülürken, kabul ve yönetişim sorumlulukları edinen tarafta toplanır. Karma model yaklaşımında kurum içi ekipler ile alt yükleniciler birlikte çalışır; bu durumda süreç sorumluluklarının ve iş ürünlerinin sahipliğinin net biçimde tanımlanması, koordinasyon riskini azaltan temel gerekliliklerden biri hâline gelir. Benzer şekilde yazılımın donanımla sıkı bütünleştiği sistemlerde (ör. gömülü

yazılım/firmware) gereksinim yönetimi, konfigürasyon yönetimi ve doğrulama faaliyetlerinin daha sıkı kontrol mekanizmalarıyla yürütülmesi beklenir; çünkü saha güncellemeleri, geri dönüş (rollback) ve test kapsama genişliği gibi konuların maliyeti artmaktadır.

ISO/IEC 12207:1995'in önemli bir özelliği, tek bir proje yaşam döngüsü modeli dayatmamasıdır. Standart, waterfall ya da çevik yaklaşımlardan birini zorunlu kılmak yerine, hangi model seçilirse seçilsin yaşam döngüsü boyunca "her koşulda yönetilmesi gereken" işleri süreç disiplini altında tanımlar. Bu yaklaşım, kuruma kendi yöntemini seçme esnekliği verirken; süreç sorumluluklarının, çıktılarının ve kontrol mekanizmalarının tutarlı biçimde standardize edilmesini mümkün kılar.

3.2 Süreç Yaklaşımı: Süreç–Aktivite–Görev Hiyerarşisi

ISO/IEC 12207:1995'in merkezindeki yaklaşım, yazılım yaşam döngüsünün yönetilebilirliğini artırmak için işin hiyerarşik bir yapı altında parçalanmasıdır. Bu hiyerarşide "süreç", belirli bir amacı gerçekleştiren ve ölçülebilir çıktılar üreten faaliyetler bütününü ifade eder; süreçler, daha somut yürütme adımları olan "aktiviteler"e; aktiviteler ise operasyonel düzeyde uygulanabilen "görevler"e ayrıştırılır. Bu yapı, yazılım mühendisliğinde sıklıkla karşılaşılan belirsiz sorumluluk alanlarını netleştirerek; planlama, izleme, raporlama ve denetim için uygulanabilir bir temel oluşturur.

Bu yaklaşımın pratikteki temel katkısı, organizasyonun kendi yaşam döngüsü modelini seçebilmesiyle birlikte, seçilen modelin "dağınık" veya kişiye bağlı bir uygulamaya dönüşmesini engellemesidir. Örneğin çevik bir yöntemde iterasyonlar üzerinden ilerlenirken dahi gereksinimlerin nasıl yönetileceği, tasarım kararlarının nasıl kayıt altına alınacağı, doğrulama kanıtlarının nasıl üretileceği ve değişikliklerin nasıl kontrol edileceği süreç yaklaşımıyla tanımlanabilir. Benzer biçimde daha ardışıl bir modelde, faz geçiş kriterleri belirlenerek teslimat riskleri azaltılabilir. Bu nedenle 12207, yöntemi belirlemekten çok, yöntemin uygulanmasında gerekli olan minimum disiplin setini ortaya koyan bir referans olarak değerlendirilebilir.

Süreç–aktivite–görev ayrıştırması aynı zamanda "uyarlama" için de işlevseldir. Proje kritiklik düzeyi, organizasyon olgunluğu, regülasyon baskısı ve sözleşme modeli gibi değişkenlere göre her süreç aynı derinlikte uygulanmayabilir; ancak seçilen süreç/aktivite/görev setinin açıkça tanımlanması ve bu seçimin gerekçelendirilmesi gerekir. Bu sayede, hem gereksiz dokümantasyon yükü oluşturulmadan yönetilebilirlik sağlanır hem de proje boyunca hangi iş ürünlerinin üretilmesi gerektiği netleşir.

3.3 Roller ve Sorumluluklar

ISO/IEC 12207:1995'in süreç yaklaşımı, yaşam döngüsündeki faaliyetlerin tek bir ekip tarafından değil; farklı sorumluluklara sahip paydaşlar tarafından yürütüleceği varsayımına dayanır. Bu nedenle rollerin açık biçimde tanımlanması, yalnızca organizasyonel düzen açısından değil; aynı zamanda süreç çıktılarının sahipliği, kabul kriterlerinin uygulanması ve denetlenebilirliğin sağlanması açısından da belirleyicidir. Standart bağlamında yaşam döngüsünde öne çıkan roller aşağıdaki şekilde konumlandırılabilir:

- **Edinen (Acquirer):** İhtiyacı tanımlar; gereksinim ve kabul ölçütlerini oluşturur; tedarik yöntemini belirler; teslimatları kabul eder ve yaşam döngüsü boyunca beklentilerin yönetimini üstlenir.

- **Tedarikçi (Supplier):** Sözleşme kapsamındaki ürün/hizmeti planlar, geliştirir/temin eder, teslim eder; durum raporlaması yapar ve kabul süreçlerinde gerekli kanıtları sunar.

- **Geliştirici (Developer):** Gereksinimlerin analizi, tasarım, kodlama, entegrasyon, test ve teslimata hazırlık faaliyetlerini yürütür; teknik iş ürünlerini üretir ve doğrulama kanıtlarını sağlar.

- **İşletmeci (Operator):** Canlı ortamın işletilmesi, hizmet sürekliliği, kullanıcı desteği, operasyon kayıtları (log/olay kaydı) ve işletim prosedürlerinin uygulanmasından sorumludur.

- **Bakımcı (Maintainer):** Hata düzeltme, iyileştirme ve uyarlama değişikliklerini; etki analizi, değişiklik kontrolü, regresyon doğrulama ve sürümleme disiplinleri altında yönetir.

Uygulamada tek bir kişi veya ekibin birden fazla rolü üstlenmesi mümkündür; ancak bu durumda dahi kritik olan nokta, rollerin sorumluluk sınırlarının yazılı biçimde tanımlanması ve süreç çıktıları üzerinden izlenebilir kılınmasıdır. Rol belirsizliği; gereksinimlerin sahipsiz kalmasına, kabul kriterlerinin netleşmemesine, değişikliklerin kontrol dışına çıkmasına ve doğrulama kanıtlarının eksik üretilmesine yol açarak hem kaliteyi hem de teslimat performansını olumsuz etkileyebilir. Bu nedenle 12207 çerçevesinde rol–süreç–çıktı ilişkisi; sözleşme metinleri, proje planı ve konfigürasyon yönetimi kayıtları gibi araçlarla kurumsal olarak güvence altına alınmalıdır.

Bu rol temelli disiplin, raporun ilerleyen bölümlerinde ele alınan ISO/IEC/IEEE 15026 yaklaşımıyla birlikte değerlendirildiğinde ayrıca anlam kazanır. Zira 15026 ailesi, özellikle assurance case kapsamında “üst seviye iddiaların” kanıtlarla gerekçelendirilmesini hedefler; bu hedefin gerçekleştirilebilmesi için kanıt üreten iş ürünlerinin sahipliğinin açık biçimde belirlenmesi gerekir. ISO/IEC/IEEE 15026-2:2022, assurance case yapısı ve terminolojisi için gereksinimler tanımlayarak güvence argümantasyonunun tutarlılığını güçlendirmeyi amaçlamaktadır.

3.4 Süreç Mimarisi: Birincil / Destek / Kurumsal

ISO/IEC 12207:1995 standardı, yazılım yaşam döngüsü boyunca gerçekleştirilen faaliyetleri üç tamamlayıcı süreç sınıfı altında ele alarak, yazılım mühendisliği çalışmalarının yalnızca teknik üretimden ibaret olmadığını; planlama, kontrol, güvence ve sürdürülebilirlik boyutlarıyla birlikte yönetilmesi gerektiğini vurgular. Bu çerçevede birincil süreçler (edinim,

tedarik, geliştirme, işletim, bakım), yazılım ürününün ortaya çıkarılmasını ve kullanım süresince değer üretmeye devam etmesini sağlayan çekirdek faaliyetleri temsil eder. Destek süreçleri (dokümantasyon, konfigürasyon yönetimi, kalite güvencesi, doğrulama, geçerleme, ortak gözden geçirme, denetim, problem çözümü) birincil süreçlerin "kontrollü ve izlenebilir" biçimde yürütülmesini sağlayan mekanizmaları kurar. Kurumsal süreçler (yönetim, altyapı, iyileştirme, eğitim) ise organizasyonun proje bazlı üretim kapasitesini kalıcı hale getirerek süreç olgunluğunu ve kurumsal öğrenmeyi destekler.

Bu mimarinin temel avantajı, yaşam döngüsü boyunca ortaya çıkan riskleri farklı düzeylerde ele alabilmesidir. Birincil süreçler, projenin "ne ürettiği ve nasıl teslim ettiği" sorusuna yanıt verirken; destek süreçleri "üretilenin nasıl doğrulandığı, değişikliklerin nasıl kontrol edildiği ve hataların nasıl yönetildiği" sorularına yanıt üretir. Kurumsal süreçler ise "bu üretimin tekrar edilebilir, ölçeklenebilir ve sürdürülebilir biçimde nasıl yürütüleceği" konusunu kurumsal yönetişim düzeyine taşır. Bu yaklaşım, özellikle çok paydaşlı ve sözleşme temelli projelerde taraflar arasında ortak bir çalışma düzeni kurmayı kolaylaştırır; çünkü süreçlerin çıktıları ve sorumlulukları tanımlandıkça kabul, denetim ve raporlama faaliyetleri somut kanıtlar üzerinden yürütülebilir.

Süreç sınıfları arasında tek yönlü bir hiyerarşi değil, karşılıklı beslenme ilişkisi bulunur. Örneğin geliştirme süreci boyunca üretilen kaynak kod, test raporları ve sürüm paketleri "birincil çıktı" niteliği taşırken; bu çıktılar, konfigürasyon yönetimi altında sürümlenmediği ve kalite güvencesi tarafından gözden geçirilmediği sürece kurumsal açıdan güvenilir bir teslimata dönüşmez. Benzer şekilde işletim sürecinde oluşan olay kayıtları, bakım sürecinin girdisi hâline gelir; bakım sürecinde yapılan değişikliklerin etkisi ise doğrulama/geçerleme ve denetim süreçlerinde yeniden değerlendirilir. Dolayısıyla 12207'nin süreç mimarisi, yaşam döngüsünde "geri besleme" ve "sürekli kontrol" ilkesini doğrudan destekleyen bir yapı kurar.

Bu mimari aynı zamanda uyarlama için de doğal bir zemin sağlar. Her projede tüm süreçlerin aynı derinlikte uygulanması hem pratik hem de maliyet açısından verimli değildir. Buna karşın; hangi süreçlerin hangi kapsamda uygulanacağı, hangi iş ürünlerinin üretileceği ve hangi kontrol noktalarının (baseline, gözden geçirme, denetim, kabul) zorunlu olacağı açıkça tanımlanmadığında proje, kişisel tercihlere bağlı yürütülen dağınık bir yapıya dönüşebilir. Bu nedenle süreç mimarisi, proje başlangıcında "asgari süreç seti" ve "asgari kanıt seti" tanımlamayı; proje ilerledikçe bu setleri değişiklik yönetimi altında güncellemeyi mümkün kılar.

Uygulama açısından süreç mimarisi değerlendirilirken aşağıdaki ilkeler belirleyici kabul edilir:

- Süreçler, yalnızca faaliyet listesi olarak değil; girdiler–faaliyet mantığı–çıktılar–kontrol noktaları bütünlüğünde ele alınmalıdır.
- Her süreç için üretilen iş ürünlerinin sahibi (sorumlu rol), sürüm durumu kabul kriteri ve izlenebilirlik bağlantısı (hangi gereksinimi/iddia kümesini destekler?) tanımlanmalıdır.

- Süreç sınıfları arası geçişlerde devir teslim paketleri ve “operasyonel yeterlilik” ölçütleri belirlenmelidir.

Bu çerçeve, raporun devamındaki bölümlerde ele alınan birincil süreçlerin her birini “amaç–girdi–temel faaliyet–çıktı” düzleminde ayrıntılandırmayı ve süreçlerin birlikte işletildiği kontrol mimarisini görünür kılmayı amaçlamaktadır.

3.5 Birincil Süreçler (Primary Processes)

ISO/IEC 12207:1995 standardında birincil süreçler, yazılım veya yazılım içeren sistemin yaşam döngüsü boyunca doğrudan değer üreten ve ürünün ortaya çıkmasını sağlayan temel faaliyetleri kapsar. Edinim, tedarik, geliştirme, işletim ve bakım süreçleri; bir yazılımın yalnızca üretilmesini değil, aynı zamanda kabul edilmesini, kullanılmasını ve değişen ihtiyaçlar altında sürdürülebilir biçimde evrilmesini hedefleyen bütünleşik bir akış oluşturur. Birincil süreçlerin etkin yürütülmesi; çoğunlukla destek süreçlerinin (özellikle konfigürasyon yönetimi, kalite güvencesi, doğrulama/geçerleme ve problem çözümü) disiplinli uygulanmasına bağlıdır.

Aşağıda her bir birincil süreç; kapsam, uygulama mantığı, tipik iş ürünleri, kontrol noktaları ve yönetimsel/teknik riskler bağlamında ayrıntılı biçimde açıklanmaktadır.

3.5.1 Edinim Süreci (Acquisition Process)

Edinim sürecinin amacı, edinen tarafın yazılım veya sistem ihtiyacını açık, doğrulanabilir ve yönetilebilir şekilde tanımlamasını; bu ihtiyacı karşılayacak çözümün uygun teknik, idari ve sözleşmesel koşullar altında seçilmesini ve kabul edilmesini sağlamaktır. Edinim, yalnızca “satın alma” faaliyeti değildir. Başarılı bir edinim; gereksinimlerin olgunlaştırılması, risklerin görünür kılınması, kabul ölçütlerinin ölçülebilir hâle getirilmesi ve tedarik ilişkisinin yönetişim mekanizmalarıyla yönetilmesini kapsar.

Bu süreçte girdiler; iş hedefleri, kullanıcı ihtiyaçları, yürürlükteki regülasyonlar, kurumsal politika ve güvenlik kuralları, zaman/bütçe kısıtları ve performans–erişilebilirlik–gizlilik gibi kritik nitelik beklentilerinden oluşur. Bu girdilerin en kritik yönü, gereksinimlerin yalnızca “ne isteniyor?” sorusuna cevap vermesi değil, aynı zamanda “ne şartta kabul edilecek?” sorusunu da yanıtlayacak şekilde kabul kriterleri üretmesidir. Kabul kriteri tanımlanmamış gereksinimler, tedarikçi değerlendirmesinde yanıltıcı tekliflerin seçilmesine; teslimat aşamasında ise kapsam anlaşmazlıklarına neden olabilir.

Edinim sürecinde tipik uygulama mantığı şu çizgide ilerler:

- Çözülmesi hedeflenen problem tanımlanır ve başarı ölçütleri belirlenir.
- İşlevsel ve işlevsel olmayan gereksinimler, test edilebilirlik ve izlenebilirlik gözetilerek netleştirilir.
- İç geliştirme, dış tedarik veya hibrit yaklaşım seçenekleri; maliyet, zaman, yetkinlik ve risk profili açısından karşılaştırılır.

- Teklif çağrısı aşamasında tedarikçiler; teknik yeterlilik, önceki proje performansı, ürün/servis olgunluğu, güvenlik yaklaşımı, teslimat kapasitesi ve toplam sahip olma maliyeti gibi ölçütlerle değerlendirilir.
- Seçim sonrasında sözleşme koşulları, teslimat planı, raporlama ve değişiklik yönetimi mekanizmaları netleştirilir; teslimatlar ise kabul planına uygun şekilde doğrulanarak devralınır.

Edinim sürecinde beklenen iş ürünleri arasında; gereksinim paketi (kabul kriterleriyle birlikte), tedarikçi değerlendirme ve seçim raporu, sözleşme/iş kapsamı dokümanları, kabul planı ve kabul kayıtları, risk listesi ve risk yönetim planı yer alır. Bu iş ürünleri, ilerleyen süreçlerde (özellikle geliştirme ve kabul) "tek referans" olarak kullanılacağından, konfigürasyon yönetimi altında sürümlenmesi ve değişikliklerin kontrollü yönetilmesi önemlidir.

Edinim sürecinin başlıca riskleri; gereksinim belirsizliği, ölçülemeyen kabul kriterleri, eksik paydaş katılımı, gerçekçi olmayan takvim/bütçe varsayımları ve tedarikçi yetkinliğinin yanlış değerlendirilmesidir. Bu riskleri azaltmak için edinim sürecinde; gereksinim gözden geçirmeleri, risk odaklı önceliklendirme ve sözleşmeye bağlı yönetişim ritimleri (düzenli raporlama, ortak gözden geçirme toplantıları) kurulması beklenir.

3.5.2 Tedarik Süreci (Supply Process)

Tedarik sürecinin amacı, tedarikçi tarafın sözleşme kapsamında üstlendiği ürün veya hizmeti planlı, kontrollü ve izlenebilir biçimde üretmesini ve teslim etmesini sağlamaktır. Tedarik, yalnızca teknik geliştirme faaliyetlerini değil; proje yönetimi, raporlama, değişiklik taleplerinin yönetimi, ara teslimatlar ve kabul hazırlığını da kapsar. Bu yönüyle tedarik süreci, müşteri beklentilerinin karşılandığını gösteren kanıtların üretilmesini ve sözleşme yönetiminin sağlıklı yürütülmesini güvence altına alır.

Tedarik sürecinin başlıca girdileri; sözleşme şartları, tanımlı proje kapsamı, teslimat takvimi, kalite ve kabul koşullarıdır. Tedarikçi, bu girdilere dayanarak kapsamı ayrıntılı biçimde yorumlar, iş kırılım yapısını oluşturur ve kaynak planlaması yapar. Planlama aşaması; teslimatların hangi sürümleme mantığıyla yapılacağını, hangi kilometre taşlarında hangi iş ürünlerinin sunulacağını ve raporlama mekanizmalarının nasıl işleyeceğini belirler. Tedarik sürecinde raporlama yalnızca idari bir yük değil, proje riskinin erken tespiti için kritik bir yönetişim aracıdır. Bu nedenle ilerleme raporları; kapsam gerçekleşmesi, riskler, açık problemler, değişiklik talepleri ve kalite göstergeleri ile birlikte sunulmalıdır.

Proje ilerledikçe ara teslimatlar gerçekleştirilir ve edinen tarafla ortak gözden geçirmeler yapılır. Bu ara teslimatlar, kabul sürecinde "son dakika sürprizlerini" azaltır; aynı zamanda gereksinim yorum farklarının erken aşamada giderilmesini sağlar. Sözleşme değişiklikleri ise zaman, maliyet, kalite ve risk üzerindeki etkileri analiz edilerek yönetilmelidir. Değişikliklerin kontrolsüz biçimde kabul edilmesi, plan sapmalarının temel nedenlerinden biridir; bu nedenle her değişiklik talebi için etki analizi (teknik etki + test/regresyon etkisi + teslimat etkisi) üretilmesi beklenir.

Tedarik sürecinin çıktıları; proje planı, ilerleme/durum raporları, ara teslim paketleri, değişiklik kayıtları, risk güncellemeleri ve kabul için gerekli kanıt setidir. Bu kanıt seti; test raporları, inceleme kayıtları, sürüm notları ve kullanım/kurulum dokümantasyonunu içerebilir. Tedarik sürecinin başarısı, teslim edilen "ürün" kadar; teslimatın izlenebilirliği ve kabul edilebilirliği açısından üretilen iş ürünlerinin kalitesiyle de ölçülür.

3.5.3 Geliştirme Süreci (Development Process)

Geliştirme sürecinin amacı, tanımlanmış gereksinimlerden hareketle yazılımın tasarlanması, gerçekleştirilmesi, test edilmesi ve teslim edilmesini sağlamaktır. Bu süreç, yazılımın teknik doğruluğu kadar; kalite, güvenlik ve performans gibi niteliklerinin de planlı biçimde güvence altına alınmasını hedefler. Geliştirme, yaşam döngüsü içinde **en görünür süreç** olmakla birlikte, çıktılarının kabul edilebilir hâle gelmesi; gereksinim yönetimi, konfigürasyon yönetimi ve doğrulama/geçerleme disiplinlerinin birlikte işletilmesini zorunlu kılar.

Geliştirme sürecinin girdileri; işlevsel ve işlevsel olmayan gereksinimler, mimari kısıtlar, uygulanacak standartlar, teknoloji seçimleri, güvenlik ve uyumluluk gereksinimleri ve kabul kriterleridir. Bu girdilerin en kritik niteliği, "test edilebilirlik" ve "izlenebilirlik" özelliklerini taşımasıdır. Gereksinimlerin test edilebilir yazılmadığı durumlarda doğrulama aşaması ölçülebilir bir sonuca bağlanamaz; izlenebilirliğin olmadığı durumlarda ise değişikliklerin etkisi yönetilemez ve regresyon riski artar. Bu nedenle geliştirme sürecinin başlangıcında gereksinimlerin çelişkisiz, önceliklendirilmiş ve izlenebilir hâle getirilmesi beklenir.

Geliştirme süreci tipik olarak tasarım kararlarının kademeli olgunlaştırılmasıyla ilerler. Üst seviye tasarım ve mimari aşamasında modüler yapı, bileşen sınırları, arayüzler, veri akışları ve kritik teknik kararlar belirlenir. Bu kararların gerekçeleri kayıt altına alınmadığında, proje ilerledikçe ortaya çıkan değişiklik talepleri karşısında tasarım bütünlüğü zayıflar. Bu nedenle mimari kararların "neden bu seçim?" sorusunu yanıtlayacak şekilde belgelenmesi, bakım sürecinde de yeniden kullanım değeri taşır.

Detay tasarım aşamasında bileşen düzeyinde algoritmalar, veri yapıları, hata yönetimi kuralları, loglama stratejisi ve bileşenlerin test edilebilirliği netleştirilir. Bu aşamada tasarımın yalnızca fonksiyonel gereksinimleri değil, aynı zamanda performans ve güvenlik gibi nitelikleri karşılayacak biçimde şekillendirilmesi gerekir. Örneğin performans hedefi olan bir sistemde veri erişim stratejilerinin, önbellekleme kararlarının ve kapasite varsayımlarının tasarım dokümanında açıkça yer alması; sonraki test aşamalarında kabul kriterlerine bağlanmayı kolaylaştırır.

Kodlama aşamasında kod standartları uygulanır, kod incelemeleri gerçekleştirilir ve birim testleri yürütülür. Kod incelemesi, yalnızca "hata yakalama" değil; güvenlik açıklarının önlenmesi, tasarım bütünlüğünün korunması ve teknik borcun kontrolü açısından da kritik bir kalite kapısıdır. Birim testleri, bileşenlerin doğruluğunu göstermekle birlikte aynı zamanda regresyon riskini azaltır; özellikle bakım sürecinde yapılan değişikliklerin yan etkilerini tespit etme kapasitesini artırır.

Bileşenlerin entegrasyonu sonrasında entegrasyon testleri ve sistem testleri yürütülür. Entegrasyon aşamasının temel riski, arayüz uyumsuzlukları ve bağımlılık sorunlarıdır; bu nedenle entegrasyon sırasının risk bazlı planlanması ve arayüz sözleşmelerinin net tanımlanması gerekir. Sistem testleri ise yalnızca işlevlerin doğruluğunu değil, işlevsel olmayan gereksinimlerin sağlandığını da gösterecek şekilde planlanmalıdır. Performans, güvenlik, yük altında kararlılık, hata toleransı ve erişilebilirlik gibi ölçütler; mümkün olduğunca ölçülebilir test senaryoları ve raporlarla kanıtlanmalıdır.

Süreç, sürümleme ve teslimat faaliyetleriyle tamamlanır. Teslimat paketi; yalnızca çalışır yazılımı değil, kurulum rehberi, sürüm notları, bilinen sorunlar listesi, işletim prosedürleri ve gerekiyorsa geri dönüş planını da kapsamalıdır. Böylece geliştirme sürecinin çıktıları, işletim sürecinde yönetilebilir bir hizmete dönüşür.

Geliştirme sürecinin tipik çıktıları arasında; gereksinim dokümanları ve izlenebilirlik kayıtları, mimari ve tasarım dokümanları, kaynak kod ve dağıtım çıktıları, test planları ve raporları, sürüm paketleri ve sürüm notları bulunur. Bu çıktılar, teslimatın kabul edilebilirliğini sağlayan kanıt setini oluşturduğundan, yapılandırma öğesi olarak ele alınmalı ve konfigürasyon yönetimi altında sürümlenmelidir.

3.5.4 İşletim Süreci (Operation Process)

İşletim sürecinin amacı, yazılım ürününün hedef ortamda kesintisiz, güvenilir ve izlenebilir biçimde çalıştırılmasını; kullanıcı desteğinin sağlanmasını ve operasyonel sürekliliğin korunmasını sağlamaktır. İşletim, geliştirme tarafından teslim edilen sürümü hizmete dönüştüren süreçtir. Bu dönüşümün başarılı olabilmesi için kurulum prosedürleri, konfigürasyon parametreleri, izleme ve loglama politikaları, yedekleme stratejileri ve olay yönetimi akışının önceden tanımlanmış olması gerekir.

İşletim sürecinin girdileri; çalışır durumdaki yazılım sürümü, kurulum talimatları, operasyon prosedürleri ve hizmet seviyesi hedefleridir. Bu girdiler, operasyonun planlı yürütülmesini sağlarken aynı zamanda hizmet kalitesinin ölçülebilir biçimde izlenebilmesine olanak tanır. Kurulum ve devreye alma aşamasında temel hedef, ortamın tekrar üretilebilir biçimde kurulması ve devreye alma sonrası doğrulama adımlarının uygulanmasıdır. Devreye alma planında geri dönüş yaklaşımı yer almadığında, özellikle kritik sistemlerde küçük bir hatanın uzun süreli kesintiye dönüşme riski artar.

Günlük işletimde izleme ve log yönetimi kritik rol oynar. Loglar; olay analizi, güvenlik denetimi ve bakım sürecinin girdisi açısından değerlidir; ancak logların aşırı üretilmesi veya düzensiz formatta tutulması, operasyonel analiz kapasitesini düşürür. Bu nedenle log seviyeleri, log saklama süreleri, kişisel veri içerebilecek alanların maskeleme kuralları ve erişim kontrolleri işletim prosedürlerinin parçası olmalıdır. Yedekleme ve geri yükleme faaliyetleri de yalnızca “yedek almak”la sınırlı değildir; geri yükleme testleriyle birlikte ele alınmadığında sahte bir güven duygusu oluşturabilir. Bu nedenle işletim sürecinde belirli periyotlarda geri yükleme denemeleri ve felaket kurtarma senaryoları çalıştırılması beklenir.

İşletim sürecinde olay ve arıza yönetimi ile kullanıcı destek süreçleri işletilir. Olayların yalnızca kapatılması değil, kök neden analiziyle tekrarının azaltılması önemlidir. Bu nedenle olay kayıtları; olayın sınıfı (kesinti, performans, güvenlik, veri sorunu vb.), etki alanı, müdahale süresi, geçici çözüm ve kalıcı çözüm aksiyonu gibi alanları içermelidir. Performans, erişilebilirlik ve kapasite gibi ölçütler düzenli izlenir; eğilim analiziyle darboğazlar erken tespit edilmeye çalışılır.

Bu sürecin çıktıları; operasyon kılavuzları, olay ve arıza kayıtları, operasyon raporları, kullanıcı destek kayıtları ve bakım sürecine aktarılan problem/iyileştirme girdileridir. İşletim çıktılarının düzenli üretilmesi, hem hizmet kalitesini yönetmeye hem de bakım değişikliklerinin etkisini objektif biçimde değerlendirmeye olanak sağlar.

3.5.5 Bakım Süreci (Maintenance Process)

Bakım sürecinin amacı, yazılımın canlı kullanım süresi boyunca ortaya çıkan hata düzeltme, iyileştirme ve uyarlama ihtiyaçlarının sistematik biçimde yönetilmesi ve uygulanmasıdır. Bakım, çoğu projede "operasyon sonrası küçük düzeltmeler" olarak görülse de pratikte yazılım yaşam döngüsünün en uzun ve maliyetli evresi olabilir. Bu nedenle bakım faaliyetlerinin plansız yürütülmesi, ürünün zamanla kontrol edilemez hâle gelmesine ve kalite kaybına yol açar.

Bakım sürecinin girdileri; hata kayıtları, değişiklik talepleri, yeni gereksinimler, uyumluluk/regülasyon değişiklikleri ve tespit edilen güvenlik açıklarıdır. Bu girdiler farklı öncelik ve risk düzeylerine sahip olabilir. Örneğin kritik bir güvenlik açığı, hızlı müdahale gerektirirken; yeni bir özellik talebi daha geniş analiz ve planlama gerektirebilir. Bu nedenle bakım süreci, taleplerin kaydedilmesi ve sınıflandırılmasıyla başlar. Her değişiklik için etki analizi yapılarak hangi bileşenlerin etkilendiği, risklerin neler olduğu, hangi testlerin yeniden çalıştırılması gerektiği ve teslimat/dağıtım planının nasıl olacağı belirlenir.

Planlama aşamasında önceliklendirme, sürüm planları, kaynak tahsisi ve gerekiyorsa geri dönüş senaryoları oluşturulur. Uygulama aşamasında değişiklikler gerçekleştirilir, birim ve entegrasyon düzeyinde doğrulanır; ardından regresyon testleri ile yan etkiler kontrol edilir. Bakımın temel problemi, yapılan küçük görünen bir değişikliğin beklenmeyen yan etkiler doğurabilmesidir. Bu nedenle regresyon yaklaşımının sistematik olması bakım kalitesinin belirleyici unsurudur. Son olarak değişiklikler konfigürasyon yönetimi altında sürümlenerek dağıtılır ve işletimde izlenir; izleme verisi, değişikliğin başarıyla sonuçlanıp sonuçlanmadığını değerlendirmek için kullanılır.

Bakım sürecinin çıktıları; değişiklik ve hata kayıtları, etki analiz raporları, düzeltme paketleri, regresyon test raporları, sürüm notları ve gerektiğinde güncellenen dokümantasyondur. Bu çıktıların düzenli üretimi, bakımın kurumsal hafızaya dönüşmesini sağlar ve yazılımın zaman içinde sürdürülebilirliğini artırır.

3.6 Destek Süreçleri (Supporting Processes)

ISO/IEC 12207 standardında destek süreçleri; edinim, tedarik, geliştirme, işletim ve bakım gibi birincil süreçlerin "doğrudan üretim" adımları olmamakla birlikte, bu süreçlerin kontrollü, izlenebilir ve tekrarlanabilir şekilde yürütülmesini sağlayan bütünleyici faaliyetleri kapsar. Destek süreçlerinin ortak yönü, yazılımın işlevlerini doğrudan geliştirmekten ziyade; kararların gerekçelendirilmesini, iş ürünlerinin bütünlüğünü, değişikliklerin denetimini ve kalite hedeflerine uyumu güvence altına almalarıdır. Bu yaklaşım, yaşam döngüsü boyunca ortaya çıkan belirsizlikleri azaltır; projeyi kişiye bağımlı "ad-hoc" uygulamalardan çıkararak kurumsal düzeyde denetlenebilir bir yapıya taşır.

Destek süreçleri, yalnızca "doküman üretimi" veya "kontrol listesi" gibi görünen faaliyetler değildir; pratikte (i) sözleşme ve kabul uyuşmazlıklarının azaltılması, (ii) bakım maliyetlerinin kontrol altında tutulması, (iii) risklerin erken görünür kılınması ve (iv) güvence/uygunluk gerektiren ortamlarda kanıt üretimi açısından temel işlev görür. Bu nedenle destek süreçlerinin beklenen çıktıları, birincil süreçlerin çıktılarıyla birlikte konfigürasyon yönetimi altında sürümlenebilir; kalite güvencesi, doğrulama ve denetim gibi mekanizmalarla tutarlılığı doğrulanabilir biçimde ele alınmalıdır.

3.6.1 Dokümantasyon (Documentation)

Dokümantasyon sürecinin amacı, yazılım yaşam döngüsü boyunca ihtiyaç duyulan bilgilerin tutarlı bir yapı içinde üretilmesini, güncel tutulmasını, erişilebilir olmasını ve gerektiğinde kanıt olarak kullanılabilmesini sağlamaktır. Dokümantasyon, yalnızca teknik ekibin iç iletişimini kolaylaştıran bir araç değil; aynı zamanda sözleşme yönetimi, kabul süreçleri, denetimler, bakım faaliyetleri ve kurumsal öğrenme açısından kritik bir altyapıdır.

Dokümantasyonun birincil katkısı, proje bilgisinin kişilere bağımlı kalmasını engellemesidir. Gereksinimlerin hangi gerekçeyle belirlendiği, tasarım kararlarının neden alındığı, bir testin hangi kabul kriterini karşıladığı, bir değişikliğin hangi sürümde ve hangi gerekçeyle yapıldığı gibi sorular; ancak düzenli ve sürümlenebilir dokümantasyonla yanıtlanabilir. Bu nedenle dokümantasyon, özellikle bakım ve problem çözümü süreçlerinin "kanıt ve izlenebilirlik" ihtiyacını karşılar.

Dokümantasyon sürecinde tipik iş ürünleri; gereksinim dokümanları, mimari ve detay tasarım belgeleri, test planları ve test raporları, kullanıcı kılavuzları, kurulum/konfigürasyon yönergeleri, işletim prosedürleri, sürüm notları, risk kayıtları ve toplantı/karar tutanaklarıdır. Dokümanların niteliği kadar, yaşam döngüsü içinde yönetimi de belirleyicidir: dokümanların sahibi, onay akışı, sürümleme kuralları, erişim yetkileri, saklama süreleri ve güncelleme sıklığı tanımlı değilse, dokümantasyon hızla güncelliğini yitirir ve güvenilir bilgi kaynağı olmaktan çıkar.

Dokümantasyon sürecinin kontrol noktaları; gereksinim ve tasarım baselinelarının oluşturulması, önemli teslimat öncesi doküman setinin bütünlük kontrolü, değişiklik taleplerinin doküman güncellemelerine bağlanması ve denetimlerde doküman-kanıt

tutarlılığının doğrulanmasıdır. Uygulamada sık görülen zayıflık; dokümanların "teslimat sonunda" yazılması ve gerçek uygulamayı yansıtmamasıdır. Bu zayıflık, özellikle bakım aşamasında yanlış kararlar alınmasına ve gereksiz deneme–yanılma maliyetine yol açar.

3.6.2 Konfigürasyon Yönetimi (Configuration Management)

Konfigürasyon yönetimi sürecinin amacı, yazılım ürününü oluşturan bileşenlerin (kaynak kod, yapılandırma dosyaları, bağımlılıklar, test varlıkları, dokümanlar vb.) sürüm, değişiklik ve bütünlük kontrolünü sistematik biçimde sağlamaktır. Bu süreç; "hangi sürüm ne zaman üretildi, hangi değişiklikler içeriyor, hangi kanıtlar bu sürümü destekliyor ve hangi ortamda çalışıyor?" sorularını yanıtlanabilir hâle getirerek, yaşam döngüsü boyunca izlenebilirliği kurumsal düzeyde güvence altına alır.

Konfigürasyon yönetimi kapsamında ilk adım, yönetilecek yapı öğelerinin tanımlanmasıdır. Bu tanım yalnızca kaynak kodu değil, ürünün tekrarlanabilir biçimde kurulabilmesi ve doğrulanabilmesi için gerekli tüm unsurları kapsamalıdır. Ardından sürümleme stratejisi belirlenir ve belirli anlarda "temel yapı" oluşturularak onaylanmış sürüm seti değişiklik kontrolüne alınır. Baseline mantığı, teslimat ve denetim süreçleri için kritik bir referans noktasıdır; çünkü kabul edilen bir sürümün sonradan kontrolsüz biçimde değişmesi, kanıtların geçerliliğini zedeler.

Değişiklik yönetimi, konfigürasyon yönetiminin ayrılmaz parçasıdır. Değişiklik talepleri; kapsam, maliyet, takvim, risk ve test/regresyon etkisi açısından değerlendirilir; onaylanır ve planlı biçimde uygulanır. Bu aşamada "durum muhasebesi" ile yapı öğelerinin güncel durumu kayıt altına alınır; hangi değişiklik hangi sürümde yer aldı, hangi testler çalıştırıldı, hangi hatalar kapatıldı gibi bilgiler izlenebilir hâle gelir. Son olarak konfigürasyon denetimleri, tanımlanan temel yapıların doğruluğunu ve bütünlüğünü kontrol eder; sürüm paketlerinin gerçekte ilan edilen içeriği taşıyıp taşımadığını doğrular.

Konfigürasyon yönetiminde kritik kontrol noktaları; sürüm etiketleme, yayın öncesi paket bütünlüğü, bağımlılıkların sürüm uyumu, ortam konfigürasyonlarının izlenebilirliği ve kanıt setlerinin ilgili sürüme bağlanmasıdır. Bu süreç zayıf kaldığında, hata tekrar üretimi zorlaşır, "hangi sürümde sorun var?" sorusu net cevaplanamaz, regresyon riski artar ve denetimlerde güvenilirlik kaybı oluşur.

3.6.3 Kalite Güvencesi (Quality Assurance)

Kalite güvencesi sürecinin amacı, yazılım geliştirme ve işletim faaliyetlerinin belirlenen kalite hedeflerine, planlara, prosedürlere ve standartlara uygun şekilde yürütüldüğünü güvence altına almaktır. Kalite güvencesi, yalnızca ürünün son hâlini değerlendirmekten ziyade; süreçlerin doğru işletilmesini sağlayarak hataların erken aşamada önlenmesini hedefler. Bu yönüyle kalite güvencesi, "hata yakalama" kadar "hata oluşumunu engelleme" yaklaşımını da içerir.

Kalite güvencesi kapsamında süreç denetimleri ve ürün incelemeleri yapılır; kalite metrikleri düzenli izlenir ve uygunsuzluklar kayıt altına alınarak kapanışa kadar takip edilir. Bu noktada kalite güvencesinin etkinliği, yalnızca uygunsuzluk bulmakla değil, uygunsuzlukların kök nedeninin analiz edilmesi ve düzeltici/önleyici faaliyetlerin tamamlanmasıyla ölçülür. Örneğin sık tekrarlanan hatalar; gereksinim yazım standartlarının zayıf olması, kod inceleme disiplininin yetersizliği veya test kapsamının hatalı kurgulanması gibi sistemik bir probleme işaret edebilir. QA, bu tür sistemik problemleri görünür kılar ve iyileştirme süreçlerine veri üretir.

Kalite güvencesi için tipik çıktılar; kalite planı/kalite hedefleri, denetim ve inceleme raporları, metrik raporları (hata trendi, test kapsamı, teslimat uygunsuzlukları vb.), uygunsuzluk kayıtları ve aksiyon takip kayıtlarıdır. Kritik kontrol noktaları arasında; plan–gerçekleşen sapma analizi, kalite kapıları, denetim bulgularının kapanış doğrulaması ve süreçlerin belgelere "uyumlu görünüp" gerçekte uygulanmaması riskine karşı kanıt temelli kontrol yer alır.

3.6.4 Doğrulama (Verification)

Doğrulama sürecinin amacı, geliştirilen yazılım ürününün tanımlı gereksinimlere, spesifikasyonlara ve tasarım kararlarına uygun biçimde üretildiğini göstermektir. Doğrulama, klasik ifadeyle "ürünü doğru yaptık mı?" sorusuna yanıt arar. Bu kapsam, yalnızca test faaliyetleriyle sınırlı olmayıp; inceleme, analiz, statik kontrol ve gösterim gibi farklı doğrulama yöntemlerini de içerebilir.

Doğrulamada temel ilke, gereksinimlerin doğrulama yöntemiyle ilişkilendirilmesidir. Bu nedenle gereksinim–test izlenebilirliği sağlanır; her gereksinimin hangi test/analiz/inceleme yoluyla doğrulandığı açıkça gösterilir. Birim testleri, entegrasyon testleri ve sistem testleri; doğrulamanın en yaygın araçlarıdır. Statik analiz ve kod incelemeleri ise özellikle güvenlik ve bakım sürdürülebilirliği açısından erken aşamada değer üretir; çünkü çalışma zamanı hataları ortaya çıkmadan riskli örüntüler tespit edilebilir.

Doğrulama çıktıları; test planı, test senaryoları, test raporları, analiz çıktıları, hata kayıtları ve düzeltme doğrulama kayıtlarıdır. Kontrol noktaları ise; test kapsamının kabul kriterlerini karşılayıp karşılamadığı, kritik gereksinimler için yeterli kanıt üretildiği, başarısız testlerin kök nedenlerinin ele alındığı ve sürüm ile doğrulama kanıtlarının konfigürasyon yönetimi altında ilişkilendirildiğidir. Doğrulamanın zayıf yürütülmesi; sonradan maliyeti çok daha yüksek olan saha hatalarına, müşteri kabul problemlerine ve bakım yükünün artmasına neden olur.

3.6.5 Geçerleme (Validation)

Geçerleme sürecinin amacı, yazılım ürününün hedef kullanım ortamında kullanıcı ihtiyaçlarını ve iş hedeflerini karşıladığını göstermektir. Geçerleme, "doğru ürünü yaptık mı?" sorusuna yanıt verir ve doğrulamadan farklı olarak kullanıcı perspektifini ve iş bağlamını merkeze alır. Bir ürün, teknik gereksinimlere uygun üretildiği hâlde gerçek

kullanım senaryolarında beklentiyi karşılamayabilir; bu nedenle geçerleme, yaşam döngüsünde ayrı bir güvence katmanı olarak ele alınır.

Geçerleme kapsamında yürütülen faaliyetler; kullanıcı kabul testleri (UAT), pilot uygulamalar, iş senaryosu testleri, kullanılabilirlik değerlendirmeleri, saha geri bildirimleri ve operasyonel kabul kontrollerini içerebilir. Geçerlemenin belirleyici unsuru, "kullanıcı ihtiyacı"nın ölçülebilir biçimde tanımlanmasıdır. Örneğin performans beklentileri yalnızca teknik metrikler değil, kullanıcı deneyimi açısından da ele alınabilir; hataların sınıflandırılması ve önceliklendirilmesi iş etkisiyle ilişkilendirilir.

Geçerleme çıktıları; UAT planı ve raporu, pilot sonuç raporları, kullanıcı geri bildirim kayıtları, iş senaryosu doğrulama sonuçları ve kabul tutanaklarıdır. Kontrol noktaları; geçerleme kapsamının gerçek kullanım senaryolarını temsil etmesi, kabul kriterlerinin iş hedefiyle uyumlu olması ve geçerleme bulgularının değişiklik yönetimi altında ele alınmasıdır. Bu süreç zayıf kaldığında ürün "teknik olarak doğru" olsa bile kullanıcı memnuniyetsizliği, iş süreçlerinde beklenmeyen aksaklıklar ve kapsam genişlemesi görülebilir.

3.6.6 Ortak Gözden Geçirme (Joint Review)

Ortak gözden geçirme sürecinin amacı, edinen taraf, tedarikçi ve proje ekibi gibi paydaşların belirli aralıklarla bir araya gelerek projenin durumunu değerlendirmesini ve ortak karar alma mekanizmasının işletilmesini sağlamaktır. Bu süreç, projenin yalnızca teknik ilerlemesini değil; kapsam, risk, kalite, değişiklik talepleri ve kabul koşulları gibi boyutlarını da eş zamanlı yönetmeyi hedefler.

Ortak gözden geçirmeler; erken geri bildirim sağlayarak sorunların "teslimat sonunda" ortaya çıkmasını önler. Toplantılarda tipik olarak; plan–gerçekleşen durumu, kritik riskler, açık hata ve problem trendleri, test durumu, sürüm planı, değişiklik talepleri ve kabul kriterlerine uyum ele alınır. Bu değerlendirme sonucunda alınan kararların kayıt altına alınması ve aksiyonların takip edilmesi, ortak gözden geçirmenin somut çıktısını oluşturur. Aksi hâlde gözden geçirmeler, yalnızca bilgilendirme toplantısına dönüşür ve risk azaltıcı etkisini kaybeder.

Ortak gözden geçirmenin çıktıları; toplantı tutanakları, karar kayıtları, aksiyon listeleri, güncellenmiş risk ve değişiklik kayıtlarıdır. Kontrol noktaları; alınan aksiyonların kapanış doğrulaması, kararların ilgili iş ürünlerine yansıtılması ve paydaşlar arasında yorum farkı oluşturabilecek belirsizliklerin giderilmesidir.

3.6.7 Denetim (Audit)

Denetim sürecinin amacı, yazılım süreçleri ve çıktılarının belirlenen standartlara, prosedürlere ve sözleşmelere uygunluğunu bağımsız veya yarı bağımsız biçimde değerlendirmektir. Denetim, projeye dışarıdan bir bakış kazandırarak; süreçlerin "kâğıt üzerinde" var olup olmadığından çok, fiilen uygulanıp uygulanmadığını ve çıktılarının yeterli kanıt niteliği taşıyıp taşımadığını değerlendirir.

Denetim sonucunda uygunsuzluklar, iyileştirme alanları ve güçlü uygulamalar belirlenir. Uygunsuzlukların yönetimi, yalnızca raporlamakla tamamlanmaz; düzeltici faaliyetlerin planlanması, uygulanması ve etkinliğinin doğrulanması gerekir. Denetim çıktıları; standartlara ve sözleşmelere uyumun kanıtı olarak kullanılabildiği gibi, süreç olgunluğunu artırmaya yönelik iyileştirme programlarına da girdi sağlar.

Denetim sürecinin tipik çıktıları; denetim planı, kontrol listeleri, denetim raporu, bulgu ve aksiyon takip kayıtlarıdır. Denetimin etkinliği; kapsamın doğru seçilmesine, denetçinin bağımsızlığına, kanıtların izlenebilirliğine ve bulguların kapanış disiplinine bağlıdır. Bu süreç zayıf kaldığında, sistemik problemler tekrarlamaya devam eder ve kurumsal öğrenme gerçekleşmez.

3.6.8 Problem Çözümü (Problem Resolution)

Problem çözümü sürecinin amacı, yaşam döngüsü boyunca ortaya çıkan hata, olay ve problemlerin kayıt altına alınmasını, analiz edilmesini, çözülmesini ve kapanışa kadar izlenmesini sistematik biçimde yönetmektir. Bu süreç yalnızca mevcut hataların giderilmesini değil, benzer problemlerin tekrarını azaltacak kök neden yaklaşımını ve trend analizini de kapsar.

Problem çözümü; genellikle problem kaydının açılması, sınıflandırma, yeniden üretme, kök neden analizi, çözüm geliştirme, doğrulama, regresyon kontrolü ve kapanış adımlarından oluşur. Bu akış içinde kritik nokta, “çözüm uygulandı” ifadesinin yeterli kabul edilmemesidir; çözümün ilgili gereksinim ve test kapsamına etkisinin değerlendirilmesi ve regresyon riskinin kontrol edilmesi gerekir. Ayrıca problem kayıtları üzerinden yapılan trend analizleri, kalite güvencesi ve süreç iyileştirme çalışmalarına doğrudan veri sağlar: belirli bir modülde tekrar eden hatalar, belirli bir ekipte sık hata üretimi veya belirli türde güvenlik açıklarının yoğunlaşması gibi göstergeler, sistemik bir iyileştirme ihtiyacına işaret eder.

Problem çözümü çıktıları; problem kayıtları, kök neden analizi notları, çözüm değişiklik paketleri, doğrulama raporları, regresyon sonuçları ve kapanış kayıtlarıdır. Kontrol noktaları; kritik problemler için müdahale süreleri, çözümün doğrulanması, tekrarlayan problemler için önleyici faaliyetlerin başlatılması ve problem verisinin kalite metriklerine entegre edilmesidir. Bu süreç kurumsal biçimde işletildiğinde, hata çözümü yalnızca “yangın söndürme” değil; uzun vadeli kalite artışını besleyen bir öğrenme mekanizmasına dönüşür.

3.7 Kurumsal Süreçler (Organizational Processes)

Kurumsal süreçler, bir yazılım organizasyonunun tek bir projeyi tamamlamasından çok daha geniş bir hedefe hizmet eder: Organizasyonun aynı kalite seviyesinde, öngörülebilir süre ve maliyetle, kişi bağımlılığına düşmeden projeleri tekrar tekrar yürütebilmesi. Bir projede geliştirme ekibi iyi olsa bile; yönetim, altyapı, iyileştirme ve eğitim süreçleri zayıfsa sonuç genellikle şuna döner: işler belirli kişilerin tecrübesiyle “idare eder”, krizlerde “kahramanlık”

yapılır, fakat kalite ve teslim tarihi tutarlılık göstermez. Bu yüzden ISO/IEC 12207, kurumsal süreçleri yaşam döngüsünün "kurumu ayakta tutan omurgası" gibi ele alır.

Kurumsal süreçler dört ana başlıkta toplanır: Yönetim (Management), Altyapı (Infrastructure), İyileştirme (Improvement), Eğitim (Training). Bu dört süreç birbirinden bağımsız değil; tam tersine birbirini besleyen bir döngü kurar. Yönetim "neye ulaşacağımızı ve nasıl ilerleyeceğimizi" belirler; altyapı bunu mümkün kılan ortam ve araçları sağlar; iyileştirme, yaşanan sorunlardan öğrenip sistemi güçlendirir; eğitim ise tüm bu yaklaşımın ekipte kalıcı yetkinliğe dönüşmesini sağlar.

3.7.1 Yönetim (Management)

Yönetim süreci, projeleri rastlantısal ilerleyen işler olmaktan çıkarıp planlı ve kontrol edilebilir hale getirmeyi amaçlar. Buradaki yönetim sadece "takvim oluşturmak" değildir; aynı zamanda kapsamın sınırlarını netleştirmek, kaynakları doğru dağıtmak, riskleri erken fark edip önlem almak ve paydaş iletişimini kurallı bir şekilde yürütmektir. Yönetim güçlü değilse projeler genellikle "başta iyi gidiyor gibi görünen ama sona doğru yetişmeyen" bir yapıya bürünür; çünkü değişiklikler kontrolsüz akar, riskler geç fark edilir ve kalite faaliyetleri sona sıkışır.

Bu süreç kapsamında tipik olarak proje başlangıcında hedefler ve başarı kriterleri belirlenir; iş paketi kırılımları çıkarılır; görev bağımlılıkları ve kilometre taşları tanımlanır; riskler sadece listelenmekle kalmayıp "sahibi ve aksiyonu" olacak şekilde yönetilir. Proje boyunca plan–gerçekleşen takibi yapılır; sapmalar erken tespit edilerek düzeltici aksiyonlar alınır. Gereksinim ve kapsam değişiklikleri geldiğinde "olur" denip geçilmez; değişikliğin zaman/maliyet/kalite etkisi değerlendirilir, resmi olarak onaylanır ve plana işlenir.

Bu yönetim disiplininin somut çıktıları genellikle proje yönetim planı, takvim ve iş kırılım dokümanları, düzenli durum raporları, risk ve issue kayıtları, değişiklik talepleri ve etki analizleri, toplantı tutanakları ve proje kapanış raporu gibi iş ürünleridir. Yönetimin olgunluğu ölçülmek istendiğinde ise teslim tarihine uyum, plan sapması, açık risk sayısı, değişiklik oranı, kritik hata trendi veya aksiyonların kapanma süresi gibi göstergeler kullanılabilir.

3.7.2 Altyapı (Infrastructure)

Altyapı süreci, ekiplerin proje yürütmek için ihtiyaç duyduğu teknik çalışma ortamını kurar ve sürdürülebilir şekilde yönetir. Burada amaç yalnızca "araç olsun" değil; ekipler arası tutarlılığı sağlayacak standart bir geliştirme/test/dağıtım düzeni oluşturmaktır. Altyapı iyi kurulmadığında projelerde çok tipik bir problem görülür: "Bende çalışıyor ama sende çalışmıyor." Bunun temel nedeni ortam farkları, yanlış sürüm bağımlılıkları, tekrarlanamayan kurulum adımları ve zayıf otomasyondur.

Bu süreç; sürüm kontrol sistemleri, derleme ve test otomasyonu (CI), dağıtım otomasyonu (CD), test ortamlarının yönetimi, izleme/loglama düzeni, dokümantasyon platformları ve erişim kontrolleri gibi unsurları kapsar. Örneğin bir kurumda "kodun ana dala birleşmeden önce otomatik testlerden geçmesi", "her sürümün izlenebilir şekilde paketlenmesi", "test

ortamının üretime benzer olması", "kritik sırların kod içinde tutulmaması" gibi kurallar altyapı sürecinin doğrudan konusudur.

Altyapının çıktıları genellikle ortam tanımları, kurulum/onboarding rehberleri, CI/CD pipeline tanımları, erişim-yetki matrisi, log/izleme standartları, yedekleme ve felaket kurtarma planları şeklinde görülür. Altyapı performansı, build başarısızlık oranı, pipeline süreleri, deployment başarı oranı, ortam kurulum süresi gibi metriklerle değerlendirilebilir.

3.7.3 İyileştirme (Improvement)

İyileştirme süreci, organizasyonun yalnızca proje bitirmesini değil, her projeden daha iyi hale gelmesini hedefler. Bu süreç, "hata oldu düzeltelim" yaklaşımını aşarak; hataların tekrar etmemesi için kök nedenleri bulmayı, süreçleri ölçmeyi, denetim bulgularını aksiyona dönüştürmeyi ve kurumun standartlarını güncellemeyi içerir. İyileştirme yoksa, aynı tip problemler farklı projelerde tekrar tekrar yaşanır: gereksinimler değişince kaos, testler geç yapılınca sürüm krizi, bakımda izlenebilirlik kaybı gibi.

Bu süreç genellikle ölçüm–analiz–aksiyon–standardizasyon–yeniden ölçüm döngüsüyle işler. Organizasyon; defect trendlerini, test kaçaklarını, değişiklik çevrim sürelerini, teslim sapmalarını düzenli izler. Sonra "neden böyle oluyor?" sorusuna kök neden analiziyle cevap arar. Ardından düzeltici/önleyici faaliyetler planlanır: örneğin kod incelemesini zorunlu yapmak, bir test katmanı eklemek, gereksinim formatını değiştirmek veya bir kontrol noktası koymak gibi. Son adımda bu iyileştirmeler kurumsal prosedürlere ve şablonlara işlenir, böylece kişilere bağlı kalmadan sürdürülebilir hale gelir.

İyileştirme çıktıları; süreç iyileştirme planları, lessons learned kayıtları, kapanan denetim bulguları raporları, güncellenmiş standartlar/şablonlar ve kök neden analizi dokümanlarıdır. Başarı ise tekrar eden hata oranının azalması, denetim bulgularının kapanma hızının artması, prod'a kaçan kritik hataların düşmesi gibi göstergelerle anlaşılır.

3.7.4 Eğitim (Training)

Eğitim süreci, kurumsal süreçlerin "kâğıt üzerinde kalmamasını" sağlayan kritik halkadır. Çünkü yönetim planları, altyapı standartları ve iyileştirme kararları ancak ekip bu yaklaşımı biliyor ve uygulayabiliyorsa işe yarar. Eğitim, yalnızca teknik eğitim anlamına gelmez; aynı zamanda süreç eğitimidir: konfigürasyon yönetimi nasıl uygulanır, değişiklik kontrolü nasıl yürür, doğrulama/validasyon çıktıları nasıl üretilir, denetime nasıl hazırlanılır gibi.

Eğitim süreci genellikle rol bazlı yetkinlik modeline dayanır. Kurum önce hangi rollerde hangi becerilere ihtiyaç olduğunu tanımlar. Ardından eğitim planı oluşturur: zorunlu eğitimler, rol bazlı eğitimler, yeni başlayanlar için onboarding programı, kritik roller için derinleşme/sertifikasyon hedefleri gibi. Eğitimlerin etkisi sadece katılım listesiyle değil; uygulamalı görevler, kısa değerlendirmeler ve proje içi performans gözlemleriyle ölçülür. Eğitim çıktıları; yetkinlik matrisi, eğitim planı, onboarding kontrol listeleri, eğitim materyalleri ve değerlendirme kayıtlarıdır.

Eğitim iyi yönetildiğinde yeni başlayanların üretkenliğe geçiş süresi kısalır, kod inceleme bulguları azalır, kritik bilgi belirli kişilerde toplanmaz ve süreç uyumu gerçek anlamda yükselir.

3.8 İzlenebilirlik, Kalite Kontrol ve Yaşam Döngüsü Boyunca Tutarlılık

ISO/IEC 12207:1995 yaklaşımında yazılım yaşam döngüsünün en kritik hedeflerinden biri, geliştirilen ürünün zaman içinde tutarlı, izlenebilir ve kontrol edilebilir kalmasını sağlamaktır. Tutarlılık, yalnızca teknik doğruluğun korunması anlamına gelmez; aynı zamanda alınan kararların, tanımlanan gereksinimlerin, üretilen çıktılarının ve yapılan değişikliklerin yaşam döngüsü boyunca birbirleriyle anlamlı ve izlenebilir ilişkiler içinde kalmasını ifade eder. Standart bu tutarlılığı, tek bir mekanizmaya değil; birbirini tamamlayan üç temel yapının birlikte işletilmesine dayandırır: dokümantasyon, konfigürasyon yönetimi ve sistematik gözden geçirme/denetim ile doğrulama–geçerleme faaliyetleri.

Dokümantasyon, tutarlılığın ilk ve en temel bileşenidir. Yazılım geliştirme sürecinde alınan kararlar, belirlenen gereksinimler, yapılan varsayımlar ve kabul edilen kısıtlar yazılı olarak kayıt altına alınmadığında, proje bilgisi hızla kişilere bağımlı hâle gelir. Bu durum özellikle ekip değişikliklerinde, bakım faaliyetlerinde veya uzun soluklu projelerde ciddi belirsizlikler doğurur. ISO/IEC 12207 çerçevesinde dokümantasyon; yalnızca “bilgi saklama” amacıyla değil, yaşam döngüsü boyunca başvurulacak yetkili referans kaynağı olarak ele alınır. Gereksinim dokümanları, tasarım kararları, test planları ve kabul kayıtları, hangi kararın hangi gerekçeyle alındığını ve hangi çıktının hangi beklentiyi karşıladığını açık biçimde ortaya koymalıdır.

Tutarlılığın ikinci ayağını oluşturan konfigürasyon yönetimi, dokümantasyonla üretilen bu bilginin zaman içinde bozulmadan korunmasını sağlar. Konfigürasyon yönetimi sayesinde hangi sürümün hangi iş ürünlerini içerdiği, hangi değişikliklerin hangi tarihte ve hangi onayla yapıldığı, hangi test ve kabul kayıtlarının hangi sürümü desteklediği izlenebilir hâle gelir. Bu yapı, özellikle bakım ve değişiklik süreçlerinde kritik önem taşır. Çünkü yazılımın canlı kullanım süresi boyunca yapılan her değişiklik, yalnızca ilgili bileşeni değil; tasarım varsayımlarını, test kapsamını ve kabul kriterlerini de etkileyebilir. Değişiklikler sürüm ve değişiklik kontrolü altında yönetilmediğinde, hatalar zincirleme biçimde büyür; bir sorunu çözmek için yapılan müdahale, başka bir bileşende yeni bir hataya yol açabilir.

Üçüncü bileşen olan gözden geçirme, denetim ve doğrulama–geçerleme faaliyetleri ise, dokümantasyon ve konfigürasyon yönetimiyle kurulan bu yapının gerçekte çalışıp çalışmadığını kontrol eden mekanizmalardır. Gözden geçirmeler, süreçlerin ve çıktılarının erken aşamalarda değerlendirilmesini sağlayarak sapmaların büyümeden tespit edilmesine olanak tanır. Denetimler ise daha bağımsız bir bakışla, süreçlerin tanımlandığı gibi uygulanıp uygulanmadığını ve üretilen iş ürünlerinin yeterli kanıt niteliği taşıyıp taşımadığını değerlendirir. Doğrulama ve geçerleme faaliyetleri, teknik ve iş gereksinimlerinin karşılandığını ölçülebilir biçimde göstererek kalite kontrolünü somutlaştırır. Bu kontroller, yalnızca “ürün çalışıyor mu?” sorusuna değil; “üretilen çıktılar güvenilebilir mi?” sorusuna da yanıt üretir.

ISO/IEC 12207 kapsamında tutarlılık, süreçlerin varlığıyla değil; bu süreçler sonucunda üretilen iş ürünlerinin sistematik biçimde ilişkilendirilmesi ile sağlanır. Gereksinim dokümanları, tasarım çıktıları, test senaryoları, test raporları ve kabul kayıtları arasında kurulan izlenebilirlik, bir gereksinimde yapılan değişikliğin hangi tasarım bileşenlerini, hangi testleri ve hangi kabul kriterlerini etkilediğinin açık biçimde görülmesini mümkün kılar. Bu izlenebilirlik, bakım ve değişiklik süreçlerinde yapılan müdahalelerin etkisini kontrol altına alarak, zincirleme hataların ve kalite kayıplarının önüne geçilmesini sağlar. Sonuç olarak 12207 yaklaşımı, yaşam döngüsü boyunca kaliteyi "sonradan denetlenen" bir özellik olmaktan çıkarıp, sürekli kontrol edilen ve kanıtlanan bir nitelik hâline getirir.

3.9 Uyarlama (Tailoring) ve Uygunluk Mantığı

ISO/IEC 12207:1995 standardının en önemli özelliklerinden biri, her projeye aynı süreç setini ve aynı ayrıntı düzeyini zorunlu kılmamasıdır. Standart, yazılım projelerinin; kapsam, kritiklik, risk profili, regülasyon baskısı, organizasyonel olgunluk ve kaynak kısıtları açısından büyük farklılıklar gösterebileceği varsayımından hareket eder. Bu nedenle 12207, süreçlerin "tek tip" uygulanmasından ziyade, proje bağlamına uygun biçimde uyarlanmasını (tailoring) öngören esnek bir yaklaşım benimser.

Uyarlama, süreçlerin rastgele çıkarılması veya sadeleştirilmesi anlamına gelmez. Aksine, uyarlama kararlarının bilinçli, gerekçeli ve izlenebilir biçimde alınması beklenir. Sağlıklı bir uyarlama sürecinin ilk adımı, proje için geçerli olan kritiklik analizinin yapılmasıdır. Yazılımın güvenlik, emniyet, veri bütünlüğü veya hizmet sürekliliği açısından taşıdığı riskler; regülasyon gereksinimleri ve olası hata sonuçları değerlendirilmeden süreç derinliğine karar verilmesi, projeyi ya aşırı bürokratik ya da kontrolsüz bir yapıya sürükleyebilir. Örneğin güvenlik kritik bir sistemde doğrulama, denetim ve izlenebilirlik mekanizmalarının hafifletilmesi kabul edilemezken; düşük riskli bir iç araç geliştirme projesinde aynı süreçlerin daha yalın biçimde uygulanması makul olabilir.

Kritiklik analizini takiben, hangi süreçlerin zorunlu olduğu, hangilerinin daha sınırlı kapsamda uygulanabileceği belirlenir. Bu aşamada önemli olan nokta, yalnızca süreç isimlerinin seçilmesi değil; süreçlerin hangi aktiviteler ve görevler düzeyinde uygulanacağının da netleştirilmesidir. Uyarlama sonucunda ortaya çıkan süreç seti, proje planı ve kalite planı gibi temel yönetim dokümanlarında açıkça tanımlanmalıdır. Böylece proje paydaşları, hangi çıktılardan sorumlu olduklarını ve hangi kontrol noktalarının geçerli olduğunu baştan bilir.

Uyarlamanın üçüncü boyutu, yapılan seçimlerin gerekçelendirilmesidir. Hangi sürecin neden daraltıldığı, hangi kontrol mekanizmasının neden zorunlu tutulduğu veya neden hafifletildiği yazılı olarak açıklanmalıdır. Bu gerekçelendirme, yalnızca iç ekip için değil; denetim, kabul ve sözleşme değerlendirmeleri açısından da kritik öneme sahiptir. Gerekçesiz uyarlama kararları, proje ilerledikçe "neden bu yapılmadı?" sorusunun cevapsız kalmasına ve güven kaybına yol açar.

Son olarak, uyarlama yaklaşımı kanıt üretimiyle tamamlanır. Seçilen süreçlerin gerçekten uygulandığını gösterecek kayıtlar, çıktılar ve kontrol sonuçları üretilmediği sürece uyarlama yalnızca teorik bir karar olarak kalır. Bu nedenle 12207'de uygunluk, "standartta yer alan her şeyin eksiksiz uygulanması" şeklinde değil; uyarlama sonucunda belirlenen süreç setinin tutarlı ve kanıtlanabilir biçimde uygulanması olarak yorumlanır. Bu yaklaşım, hem proje esnekliğini korur hem de kalite ve denetlenebilirlikten ödün verilmesini engeller.

4. ISO/IEC 15026 — Sistem ve Yazılım Güvencesi (Assurance)

ISO/IEC 15026 standart ailesi, sistem ve yazılımın belirli kritik özellikler açısından güvenilirliğinin yalnızca "üretilmiş olması" ile değil; aynı zamanda bu özelliklerin sağlandığının gerekçelendirilmesi ve kanıtlanması ile yönetilmesi gerektiği varsayımından hareket eder. Yazılım mühendisliği uygulamalarında test, analiz, inceleme ve operasyon verisi gibi çok sayıda çıktı üretilse bile, bu çıktılar çoğu durumda dağınık kalabilir; hangi testin hangi iddiayı desteklediği, hangi analiz sonucunun hangi risk azaltımını sağladığı veya hangi varsayımlar altında bu sonuçlara güvenilebileceği açık biçimde kurulmadığında, teknik çalışmaların ikna ediciliği azalır. ISO/IEC 15026 yaklaşımı bu noktada, "kanıt üretmek" ile "kanıt üzerinden güven tesis etmek" arasındaki boşluğu kapatan bir çerçeve sunar. Özellikle güvenlik, emniyet, bütünlük, güvenilirlik ve süreklilik gibi kritik niteliklerde, paydaşların beklentisi yalnızca iyi niyetli bir mühendislik faaliyeti değil; iddia–argüman–kanıt ilişkisiyle kurulmuş sistematik bir güvence mantığıdır.

4.1 ISO/IEC 15026 Ailesinin Amacı ve Parçaları

ISO/IEC 15026 ailesinin temel amacı, sistem veya yazılımın belirli bir kritik özelliği sağladığına dair güvenin; teknik kanıtlar, mantıksal argümanlar ve açık varsayımlar üzerinden yapılandırılmış biçimde kurulmasını sağlamaktır. Bu yaklaşım, kalite yönetimi perspektifinde yalnızca "sonuç ölçmek" değil; sonuçların hangi koşullarda geçerli olduğunu ve hangi gerekçeyle ikna edici sayılacağını da tanımlamayı hedefler. Bu nedenle 15026 ailesi, güvenceyi bir "dokümantasyon faaliyeti" olarak değil; yaşam döngüsüne entegre edilen ve değişiklik yönetimiyle birlikte evrilen bir yönetişim alanı olarak ele alır.

Ailenin öne çıkan parçaları aşağıdaki şekilde özetlenebilir:

- **ISO/IEC/IEEE 15026-1 (Vocabulary and Concepts):** Güvence terminolojisini ve kavramsal temeli tanımlayan bileşen olarak değerlendirilir; güvenceye ilişkin kavram setinin tutarlı kullanılmasını amaçlar.
- **ISO/IEC/IEEE 15026-2 (Assurance Case):** Güvence dosyalarının yapı ve terminolojisine ilişkin gereksinimleri tanımlayarak, bir iddianın nasıl argümantasyon ve kanıt ile destekleneceğini çerçeveler. Katalog özetinde standardın assurance case'in yapısı ve terminolojisine yönelik gereksinimler içerdiği ve assurance case geliştirmede uygulanabilir olduğu belirtilmektedir.
- **ISO/IEC 15026-3:2011 (System/Software Integrity Levels):** Bütünlük seviyesi kavramını tanımlar ve integrity level'in sistemlere, yazılım ürünlerine, öğelere ve dış

bağımlılıklara atanmasına ilişkin gereksinim ve yöntem önerileri sunar. Böylece kritiklik arttıkça beklenen güvence yoğunluğu artırılabilir.

- **ISO/IEC/IEEE 15026-4:2021 (Assurance in the Life Cycle):** Güvence faaliyetlerini yaşam döngüsü süreçleriyle ilişkilendiren "süreç görünümü" yaklaşımını sunar. Katalog özetinde, seçilmiş bir iddianın gerçekleştirilmesi ve gerçekleştirildiğinin gösterilmesi için rehberlik verildiği ve bu rehberliğin ISO/IEC/IEEE 12207 ile ISO/IEC/IEEE 15288 üzerine inşa edildiği belirtilmektedir.

Bu parçalar birlikte ele alındığında, 15026 ailesi bir yandan güvence argümantasyonunun "nasıl yazılacağını" (15026-2), diğer yandan güvence yoğunluğunun "hangi kritiklik düzeyinde ne kadar olması gerektiğini" (15026-3) ve nihayet bu yaklaşımın "yaşam döngüsü içinde nasıl yönetileceğini" (15026-4) tanımlayan bütünleşik bir yapı sunar.

Versiyonlar:

- **ISO/IEC 15026-1:** Üç farklı versiyonu bulunmaktadır:
    - ISO/IEC TR 15026-1:2010
    - ISO/IEC 15026-1:2013
    - ISO/IEC/IEEE 15026-1:2019
- **ISO/IEC 15026-2:** İki farklı versiyonu bulunmaktadır:
    - ISO/IEC 15026-2:2011
    - ISO/IEC/IEEE 15026-2:2022
- **ISO/IEC 15026-3:** Dört farklı versiyonu bulunmaktadır:
    - ISO/IEC 15026:1998
    - ISO/IEC 15026-3:2011
    - ISO/IEC 15026-3:2015
    - ISO/IEC/IEEE 15026-3:2023
- **ISO/IEC 15026-4:** İki farklı versiyonu bulunmaktadır:
    - ISO/IEC 15026-4:2012
    - ISO/IEC/IEEE 15026-4:2021

Bu versiyonlar, güvence sistemlerinin sürekli gelişen ve çeşitli endüstri gereksinimlerine uyum sağlayan bir yapıda kalmasını sağlar. Aynı zamanda, her bir versiyonun farklı zaman dilimlerinde ve farklı endüstriyel bağlamlarda geliştirilmiş olması, ISO/IEC 15026 ailesinin esnekliğini ve adaptasyon yeteneğini yansıtmaktadır.

4.2 Güvence (Assurance) Neyi Çözer?

Güvence yaklaşımının odaklandığı temel soru şu şekilde ifade edilebilir: “Bu sistemin X özelliğini (ör. güvenlik, emniyet, güvenilirlik, bütünlük) sağladığına neden güvenmeliyiz?” Bu soru, yalnızca test çalıştırmakla veya bir değerlendirme raporu üretmekle kendiliğinden yanıtlanmış olmaz. Çünkü test sonuçlarının yorumlanması, kapsamın yeterliliği, test ortamının temsil gücü, kullanılan analiz yönteminin geçerliliği ve bu sonuçların hangi iddiayı hangi şartlar altında desteklediği çoğu zaman açıkça ifade edilmez. Bu durum “kanıt üretildiği halde ikna ediciliğin zayıf kalması” gibi tipik bir probleme yol açar: teknik ekip çok sayıda çıktı üretir; ancak paydaşın aradığı şey, bu çıktılar arasındaki mantıksal ilişki ve güven gerekçesidir.

ISO/IEC 15026 yaklaşımı, bu boşluğu kapatmak için kanıtları “sistematik bir argümantasyon” içinde örgütler. Böylece güvence, tekil test veya tekil dokümanların toplamından oluşan dağınık bir küme olmaktan çıkar; üst seviye iddiaların alt iddialara ayrıştırıldığı ve her alt iddianın uygun kanıtlarla desteklendiği bir yapı haline gelir. Bunun pratik sonucu şudur: Bir denetçi, müşteri veya düzenleyici otorite, belirli bir güvenlik/kalite iddiası için hangi kanıtların sunulduğunu, bu kanıtların hangi varsayımlar altında geçerli olduğunu ve argüman zincirinin hangi adımlarla kurulduğunu tutarlı bir çerçevede izleyebilir. Bu yaklaşım, özellikle kritik sistemlerde “güven”in kişisel kanaate değil, yapılandırılmış gerekçelendirmeye dayanmasını sağlar.

4.3 ISO/IEC 15026-2: Güvence Dosyası (Assurance Case)

ISO/IEC/IEEE 15026-2, assurance case kavramını bir “rapor biçimi” olarak sınırlandırmaktan ziyade, güvence argümantasyonunun nasıl kurulacağına ilişkin bir çerçeve olarak ele alır. Katalog özetinde standardın assurance case’in yapı ve terminolojisi için gereksinimler tanımladığı belirtilmektedir. Bu bağlamda assurance case; yalnızca mevcut test raporlarının bir araya getirilmesi değil, üst seviye bir iddiayı desteklemek üzere kanıtların hangi mantıkla seçildiğini ve nasıl bağlandığını açıklayan yapılandırılmış bir gerekçelendirme sistemidir.

4.3.1 Yapı Taşları

Assurance case’in temel yapı taşları, güvence argümantasyonunun şeffaflığını ve denetlenebilirliğini doğrudan belirler:

- **İddia (Claim):** Kanıtlanmak istenen özellik ifadesidir. İddia mümkün olduğunca açık, ölçülebilir ve sınırları belirli şekilde yazılmalıdır. “Sistem güvenlidir” gibi belirsiz ifadeler yerine “Sistem, yetkisiz erişime karşı kimlik doğrulama ve rol tabanlı yetkilendirme ile korunmaktadır” gibi kapsamı belirli iddialar tercih edilir.

- **Argüman (Argument):** İddianın neden doğru kabul edilebileceğini adım adım açıklayan mantık zinciridir. Argüman; iddiayı alt iddialara böler, risk azaltım mekanizmalarını gösterir ve her bir alt iddia için hangi kanıt türünün uygun olduğunu

temellendirir.

- **Kanıt (Evidence):** İddianın desteklenmesinde kullanılan test raporları, analiz çıktıları, inceleme kayıtları, formal doğrulama sonuçları, operasyon verileri, denetim raporları veya sertifikasyon kayıtları gibi iş ürünleridir. Kanıtın değeri; kapsam, güncellik, bağımsızlık, tekrar üretilebilirlik ve izlenebilirlik gibi niteliklerle değerlendirilir.

- **Varsayımlar (Assumptions):** Argümanın geçerliliğinin dayandığı açık kabullerdir. Örneğin "kullanılan kripto kütüphanesinin belirli bir standarda uygun olduğu" veya "kimlik sağlayıcının belirli SLA koşullarında çalıştığı" gibi varsayımlar, assurance case içinde açıkça belirtilmediğinde argüman örtük hale gelir ve ikna edicilik azalır.

Bu yapı taşlarının birlikte kullanımı, iddiayı yalnızca "sonuç bildirimi" olmaktan çıkarır; iddianın altında yatan gerekçeyi şeffaflaştırır. Standart yaklaşımı, assurance case'in çok seviyeli alt iddialar aracılığıyla kanıtlara bağlanan yapılandırılmış bir argümantasyon olduğunu ve varsayımların görünür olması gerektiğini vurgulayan bir mantıkla uyumludur.

4.3.2 Neden "Assurance Case" Belgesi Tek Başına Değil, Bir Yönetim Nesnesidir?

Assurance case'in etkili kullanımı, onun statik bir doküman olarak ele alınmamasını gerektirir. Yazılım sistemleri yaşam döngüsü boyunca değişir: yeni özellikler eklenir, mevcut bileşenler refactor edilir, güvenlik açıkları ortaya çıkabilir, bağımlılıklar güncellenir ve işletim ortamı dönüşebilir. Bu değişimlerin her biri, önceki güvence argümantasyonunun geçerliliğini etkileyebilir. Bu nedenle assurance case, ürünle birlikte yaşayan ve güncellenen "canlı" bir yapı olarak yönetilmelidir.

Yeni bir özellik eklendiğinde, sistemin risk profili değişebilir ve üst seviye iddialar için yeni alt iddialar veya yeni kanıt türleri gerekebilir. Bakım sürecinde yapılan bir düzeltme, daha önce sunulan kanıtların bir kısmını geçersiz kılabilir; örneğin güvenlik bileşeninde değişiklik yapıldıysa, önceki penetrasyon testi raporunun ilgili bölümleri artık aynı sürüm için geçerli olmayabilir. Bu nedenle assurance case, konfigürasyon yönetimi altında izlenmeli; hangi sürüm için hangi iddianın hangi kanıtlarla desteklendiği sürüm temelli olarak gösterilebilmelidir. ISO/IEC/IEEE 15026-4'ün yaşam döngüsü içinde güvenceyi ele alan yaklaşımı da, güvence faaliyetlerinin süreçlerle birlikte yürütülmesini ve yaşam döngüsü boyunca güncel tutulmasını öngören bir çizgide konumlanmaktadır.

Assurance case'in yönetim nesnesi olarak ele alınması, pratikte şu disiplinleri gerekli kılar: (i) iddiaların ve kanıtların sürümleme mantığıyla ilişkilendirilmesi, (ii) değişiklik taleplerinde assurance etkisinin (assurance impact) değerlendirilmesi, (iii) kanıt üretiminin proje planına entegre edilmesi ve (iv) periyodik gözden geçirmelerle argüman zincirinin güncelliğinin doğrulanması. Bu disiplinler işletildiğinde assurance case, yalnızca "denetim zamanı hazırlanan bir dosya" değil; karar alma süreçlerini destekleyen, riskleri görünür kılan ve kaliteyi somutlaştıran bir yönetim aracına dönüşür.

4.4 ISO/IEC 15026-3: İntegrity level (System Integrity Levels)

ISO/IEC 15026-3 standardı, sistem ve yazılım bileşenlerinin taşıdığı kritiklik düzeyine bağlı olarak farklı güvence yoğunluklarının uygulanmasını sistematik hâle getirmeyi amaçlar. ISO özetinde belirtildiği üzere standart; integrity level kavramını tanımlar, bu seviyelerin gösterimi için gerekli gereksinimleri açıklar ve sistemlere, yazılım ürünlerine, alt bileşenlere ve ilgili dış bağımlılıklara bütünlük seviyesi atanmasına yönelik yöntem önerileri sunar. Bu yaklaşım, güvence faaliyetlerinin her bileşen için aynı derinlikte yürütülmesi varsayımını terk ederek, risk ve etki temelli bir güvence mimarisi kurulmasını mümkün kılar.

Yazılım sistemleri pratikte homojen yapılardan oluşmaz. Aynı sistem içinde kullanıcı arayüzü bileşenleri, veri işleme modülleri, güvenlik mekanizmaları, dış servis entegrasyonları ve altyapı bağımlılıkları birlikte yer alır; ancak bu öğelerin hata durumunda doğuracağı sonuçlar eşit değildir. Örneğin bir raporlama ekranındaki görsel hata ile kimlik doğrulama mekanizmasındaki bir zafiyetin etkisi aynı düzeyde değildir. ISO/IEC 15026-3, bu farklılığı merkezine alarak “her bileşene aynı doğrulama ve kanıt yükünü uygulamak” yerine, kritikliği yüksek öğeler için daha güçlü güvence beklentileri tanımlanmasını önerir.

4.4.1 Integrity Level Mantığı

Bütünlük seviyesi yaklaşımının temel varsayımı, her sistem öğesinin aynı derecede kritik olmadığıdır. Bu nedenle güvence faaliyetleri, öğelerin başarısızlık durumunda yaratacağı etkiye göre farklılaştırılmalıdır. Integrity level, bu farklılaştırmayı biçimsel ve tutarlı bir yapıya oturtur. Daha yüksek bütünlük seviyesine sahip bir öğe için; daha kapsamlı doğrulama faaliyetleri, bağımsız inceleme ve denetimler, daha sıkı değişiklik kontrolü ve daha güçlü kanıt setleri beklenir. Düşük bütünlük seviyesine sahip öğelerde ise, gereksiz bürokratik yük oluşturmadan daha yalın güvence mekanizmaları yeterli olabilir.

Bu yaklaşım, yalnızca teknik doğrulama faaliyetlerinin yoğunluğunu değil; aynı zamanda süreç disiplinini de etkiler. Yüksek bütünlük seviyesine atanmış bir bileşende yapılan değişiklikler, daha ayrıntılı etki analizi gerektirir; değişiklik öncesi ve sonrası doğrulama kapsamı genişletilir ve mevcut assurance case argümantasyonu yeniden değerlendirilir. Bu durum, bakım ve evrim süreçlerinde “kritik bileşenlerin korunmasını” sistematik hâle getirir. Integrity level, böylece kaliteyi soyut bir hedef olmaktan çıkarıp, farklı risk profilleri için ölçülebilir güvence beklentilerine dönüştürür.

Bütünlük seviyesi yaklaşımının önemli bir yönü de, güvence gereksinimlerinin orantılılık ilkesine dayanmasıdır. Tüm bileşenler için en yüksek güvence seviyesini zorunlu kılmak; zaman, maliyet ve kaynak açısından sürdürülebilir değildir. ISO/IEC 15026-3, bu gerçeği kabul ederek; güvence yoğunluğunun sistematik biçimde ayarlanmasını ve bu ayarlamanın açıkça gerekçelendirilmesini bekler. Böylece güvence faaliyetleri, rastgele veya sezgisel kararlara değil; yapılandırılmış bir kritiklik değerlendirmesine dayanır.

4.4.2 Dış Bağımlılıklar Neden Dahildir?

ISO/IEC 15026-3'ün ayırt edici yönlerinden biri, bütünlük seviyesi yaklaşımını yalnızca "kurumun kendi geliştirdiği yazılım" ile sınırlamamasıdır. Modern yazılım sistemleri; üçüncü taraf kütüphaneler, açık kaynak bileşenler, harici servisler, donanım bileşenleri ve altyapı hizmetleri gibi çok sayıda dış bağımlılıkla birlikte çalışır. Bu bağımlılıklar, sistemin işlevselliği kadar güvenliği ve güvenilirliği üzerinde de doğrudan etki yaratır. Dolayısıyla sistem bütünlüğünün değerlendirilmesi, yalnızca iç bileşenlerin kalitesine bakılarak tamamlanamaz.

ISO/IEC 15026-3, dış bağımlılıkların da bütünlük seviyesi perspektifiyle ele alınmasını önererek, güvence yaklaşımını gerçekçi bir zemine oturtur. Örneğin güvenlik kritik bir bileşen, üçüncü taraf bir kripto kütüphanesine dayanıyorsa; bu kütüphanenin hangi standartlara uygun olduğu, hangi sürümünün kullanıldığı, bilinen zafiyetlerinin olup olmadığı ve güncelleme politikasının ne olduğu güvence argümantasyonunun parçası hâline gelir. Benzer biçimde harici bir servis entegrasyonu, servis sağlayıcının hizmet seviyesi anlaşmaları (SLA), erişilebilirlik garantileri ve olay yönetimi kapasitesiyle birlikte değerlendirilir.

Dış bağımlılıkların bütünlük seviyesi kapsamına alınması, güvenceyi "kontrol edilebilir alan" ile sınırlı tutmamak anlamına gelir. Her dış bağımlılık doğrudan kontrol edilemeyebilir; ancak bu durum, onların etkisinin göz ardı edilmesini meşrulaştırmaz. ISO/IEC 15026-3 yaklaşımı, bu noktada açık varsayımlar, ek kanıtlar veya risk kabul kararları üzerinden argümantasyon kurulmasını teşvik eder. Örneğin bir üçüncü taraf bileşenin belirli bir sertifikaya sahip olduğu varsayımı açıkça belirtilir; bu varsayımın geçerliliği düzenli olarak gözden geçirilir ve bağımlılığın bütünlük seviyesine uygun izleme mekanizmaları tanımlanır.

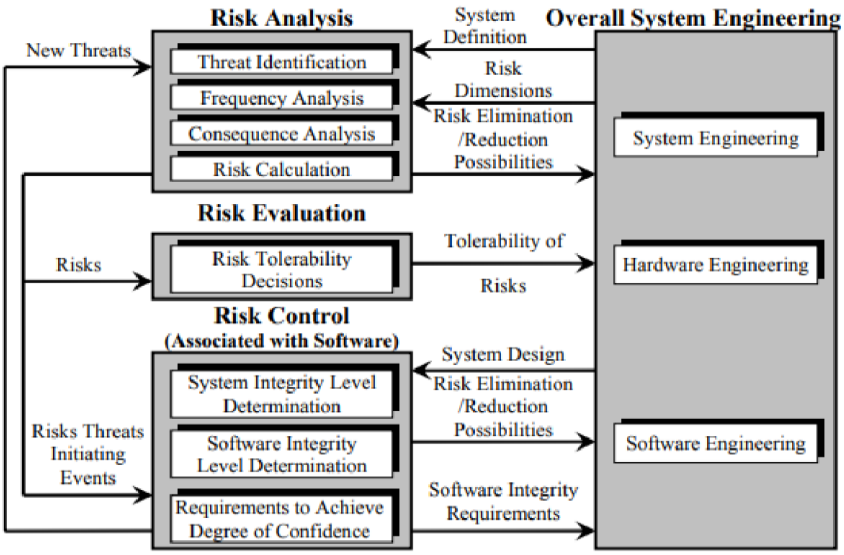
Sonuç olarak ISO/IEC 15026-3, bütünlük seviyesi kavramı aracılığıyla güvence faaliyetlerini risk odaklı, orantılı ve kapsamlı bir yapıya kavuşturur. Hem iç bileşenlerin hem de dış bağımlılıkların sistem bütünlüğü üzerindeki etkisini görünür hâle getirerek, assurance case yaklaşımının gerçek sistem karmaşıklığını yansıtmasını sağlar. Bu yaklaşım, ISO/IEC 12207 süreçlerinden üretilen iş ürünlerinin, hangi bileşenler için ne düzeyde kanıt oluşturması gerektiğini belirlemede de doğrudan yol gösterici rol oynar.

4.5 ISO/IEC/IEEE 15026-4: Assurance in the Life Cycle

ISO/IEC/IEEE 15026-4'ün ISO ve IEEE özetlerinde; seçilmiş bir iddia için güvence sağlamak üzere rehberlik sunduğu ve bunu **ISO/IEC/IEEE 15288** üzerine sistem güvence süreç görünümü, **ISO/IEC/IEEE 12207** üzerine yazılım güvence süreç görünümü olarak konumlandırdığı belirtilir.

Bu ifade pratikte şu anlama gelir:

- 15026-4, "güvence faaliyetleri"ni yaşam döngüsü süreçlerine entegre etmeyi hedefler.
- Güvence; geliştirme sürecinden kopuk bir ek doküman işi değil, sürecin ürettiği çıktıları organize eden ve yeterliliğini değerlendiren bir katmandır.



Şekil 1: Risk analizi, risk değerlendirme ve bütünlük seviyesi belirleme sürecinin sistem ve yazılım mühendisliği faaliyetleriyle ilişkisi

Şekil 1, yazılım bütünlüğü düzeyi belirleme sürecine genel bir bakış sunmakta; risk analizi, risk değerlendirme ve risk kontrol adımlarının sistem ve yazılım mühendisliği faaliyetleriyle nasıl bütünleştirildiğini göstermektedir.

5.ISO/IEC 15026'nın Sınırlamaları ve Değerlendirilmesi

ISO/IEC 15026 standart ailesi tarafından önerilen assurance case yaklaşımı, özellikle güvenlik, emniyet, bütünlük ve güvenilirlik gibi kritik niteliklerin ön planda olduğu sistemlerde yüksek düzeyde güvence sağlamayı hedefleyen güçlü bir çerçeve sunmaktadır. Bununla birlikte, bu yaklaşımın her yazılım projesi için evrensel ve sorunsuz bir çözüm olarak değerlendirilmesi mümkün değildir. Assurance case temelli güvence yaklaşımı, sahip olduğu yapısal avantajların yanı sıra, uygulama bağlamına bağlı olarak ortaya çıkan önemli sınırlamaları da bünyesinde barındırmaktadır.

Öncelikle assurance case oluşturma süreci, doğası gereği yüksek düzeyde zaman, maliyet ve uzmanlık gerektiren bir faaliyettir. Üst seviye iddiaların tanımlanması, bu iddiaların alt iddialara ayrıştırılması, uygun argüman zincirlerinin kurulması ve her bir argümanın yeterli ve geçerli kanıtlarla desteklenmesi; yalnızca teknik bilgi değil, aynı zamanda sistematik düşünme ve güvence metodolojisine hâkimiyet gerektirir. Bu durum, özellikle küçük ölçekli, kısa süreli veya düşük riskli yazılım projelerinde assurance case yaklaşımının uygulanmasını pratik olmaktan çıkarabilir. Böyle projelerde, assurance case için harcanan eforun sağlayacağı ek fayda, maliyetine kıyasla sınırlı kalabilir ve bu yaklaşım gereğinden fazla karmaşık bir yönetim yükü oluşturabilir.

İkinci önemli sınırlama, assurance case yapısının argümantasyon temelli olmasından kaynaklanır. Assurance case, teknik kanıtların yanı sıra bu kanıtları birbirine bağlayan mantıksal argümanlara dayanır. Bu argümanların kurgulanması ve değerlendirilmesi ise

kaçınılmaz olarak insan yorumuna bağlıdır. Yanlış yapılandırılmış, eksik gerekçelendirilmiş veya kritik varsayımları yeterince açık olmayan bir argümantasyon; teknik olarak yeterli görünen ancak gerçekte ciddi riskler barındıran sistemlerin "güvenli" kabul edilmesine yol açabilir. Bu durum, assurance case'in biçimsel olarak eksiksiz görünmesine rağmen, özünde yanıltıcı bir güven hissi oluşturması riskini beraberinde getirir.

Bu bağlamda assurance case'in kalitesi, yalnızca sunulan kanıtların sayısına değil; argümanların tutarlılığı, tamlığı ve eleştirel değerlendirmeye açıklığı ile doğrudan ilişkilidir. Denetim veya gözden geçirme süreçleri yeterince bağımsız ve derinlikli yürütülmediğinde, assurance case zamanla "kanıtların listelendiği" ancak risklerin gerçekçi biçimde tartışılmadığı bir belgeye dönüşebilir. Bu durum, ISO/IEC 15026 yaklaşımının yanlış uygulanması hâlinde, güvenceyi artırmak yerine güvence algısını zayıflatan bir sonuç doğurmasına neden olabilir.

Bir diğer sınırlayıcı unsur, assurance case yaklaşımının kurumsal süreç olgunluğuna olan bağımlılığıdır. ISO/IEC 15026, güvence faaliyetlerinin yaşam döngüsüne entegre edilmesini öngörür; ancak bu entegrasyonun sağlıklı biçimde yürütülebilmesi için dokümantasyon, konfigürasyon yönetimi, doğrulama–geçerleme ve değişiklik yönetimi gibi süreçlerin zaten belirli bir olgunluk seviyesinde olması gerekir. Süreç disiplini zayıf, izlenebilirlik mekanizmaları yeterince kurulmamış veya kalite güvence kültürü gelişmemiş organizasyonlarda assurance case yaklaşımı, gerçekçi bir güvence aracı olmaktan ziyade biçimsel bir yük hâline gelebilir.

Buna ek olarak, assurance case yaklaşımı dinamik ve hızlı değişen geliştirme ortamlarıyla her zaman doğal bir uyum göstermeyebilir. Özellikle sık sürüm çıkarılan, gereksinimlerin hızla evrildiği veya çevik yöntemlerin yoğun kullanıldığı projelerde assurance case'in sürekli güncel tutulması önemli bir yönetim zorluğu oluşturur. Assurance case'in her değişiklikte gözden geçirilmemesi, zamanla ürün ile güvence argümantasyonu arasında kopukluk oluşmasına neden olabilir. Bu durum, assurance case'in "yaşayan bir yapı" olma niteliğini kaybederek, geçmiş bir sürümün varsayımlarını yansıtan statik bir belgeye dönüşmesi riskini doğurur.

Tüm bu sınırlamalar dikkate alındığında, ISO/IEC 15026'nın her yazılım projesi için zorunlu veya evrensel bir çözüm olarak değerlendirilmesi uygun değildir. Bu standart ailesi, özellikle hata maliyetinin çok yüksek olduğu, düzenleyici denetimlerin bulunduğu veya insan güvenliği, veri bütünlüğü ve hizmet sürekliliği gibi kritik niteliklerin ön planda olduğu sistemler için güçlü ve anlamlı bir güvence yaklaşımı sunar. Düşük riskli ve sınırlı kapsamlı projelerde ise, assurance case yaklaşımının sadeleştirilmiş veya seçici biçimde uygulanması daha dengeli bir çözüm olabilir.

Sonuç olarak ISO/IEC 15026, doğru bağlamda ve doğru olgunluk düzeyinde uygulandığında, yazılım ve sistem güvenliğini kanıt temelli biçimde gerekçelendiren güçlü bir araçtır. Ancak bu gücün sürdürülebilir ve anlamlı olabilmesi, yaklaşımın sınırlamalarının bilinmesi, insan faktörünün farkında olunması ve assurance case'in yaşam döngüsü boyunca aktif biçimde yönetilmesiyle mümkündür. Bu nedenle ISO/IEC 15026, bir "her derde deva" standart olarak

değil; yüksek kritiklik gerektiren sistemler için bilinçli şekilde seçilmesi gereken özel bir güvence çerçevesi olarak değerlendirilmelidir.

6. ISO/IEC 12207 ile ISO/IEC 15026'nın Birlikte Kullanımı

ISO/IEC 12207 ve ISO/IEC 15026, yazılım mühendisliği yaşam döngüsüne farklı sorular üzerinden yaklaşmakta; ancak bu farklılık, standartların birbirini dışlamasından ziyade tamamlayıcı bir ilişki kurmasına olanak tanımaktadır. ISO/IEC 12207 esas olarak yazılım yaşam döngüsünde rollerin, süreçlerin ve faaliyetlerin nasıl yapılandırılacağını tanımlarken; ISO/IEC 15026, bu yapı içinde üretilen çıktılardan nasıl güvence üretileceğini ve bu güvencenin hangi argümanlar ve kanıtlar üzerinden gerekçelendirileceğini ele almaktadır. Bu bağlamda iki standart, yazılım projelerinde sıklıkla ayrı ayrı ele alınan "süreç disiplini" ile "kanıta dayalı güven" kavramlarını ortak bir çerçevede buluşturmaktadır.

ISO/IEC 12207, "Yaşam döngüsünde kim, neyi, hangi sorumlulukla yapacak?" sorusuna yanıt verirken; ISO/IEC 15026, "Belirli bir kritik özelliğin sağlandığına neden güvenmeliyiz?" sorusunu merkeze alır. Bu ayrım, standartların uygulama alanlarını netleştirmekle birlikte, aynı zamanda aralarındaki entegrasyon potansiyelini de ortaya koyar. ISO/IEC 12207 süreçleri, yazılım geliştirme faaliyetlerini planlı ve izlenebilir hâle getirerek güvence için gerekli olan ham veriyi üretir; ISO/IEC 15026 ise bu veriyi yapılandırılmış bir assurance case içinde anlamlı ve ikna edici bir güvence argümantasyonuna dönüştürür.

6.1 Kanıt Üretimi Açısından Haritalama

Aşağıdaki Tablo 2 , 12207 süreçlerinin 15026 assurance case için tipik "kanıt kaynakları" olarak nasıl çalıştığını gösterir:

12207 Süreci	Assurance Case'te Tipik Rolü	Örnek Kanıt
Development	Teknik gerçekleştirme + doğrulamaya hazırlık	Tasarım dokümanı, kod inceleme kayıtları, test raporları
Verification	"Gereksinime uygunluk" kanıtı	İzlenebilirlik matrisi, birim/entegrasyon test sonuçları
Validation	"İhtiyacı karşılama" kanıtı	UAT, pilot kullanım sonuçları
Configuration Management	Kanıtların sürüm bütünlüğü	Baseline kayıtları, değişiklik kontrol kayıtları
QA / Audit	Süreç ve ürün uygunluğu	Denetim raporları, uygunsuzluk/aksiyon takibi
Operation / Maintenance	Sahadan gerçek veri ve süreklilik	Olay kayıtları, güvenlik yamaları, regresyon raporları

Tablo 1: ISO/IEC 12207 Süreçlerinin ISO/IEC 15026 Assurance Case İçin Kanıt Kaynakları Olarak Kullanımı

Bu haritalama, ISO/IEC/IEEE 15026-4'te tanımlanan "güvence faaliyetlerinin yaşam döngüsü süreçleri üzerine oturtulması" yaklaşımıyla doğrudan uyumludur. ISO/IEC 15026-4, güvenceyi ayrı bir faaliyet kümesi olarak değil; mevcut yaşam döngüsü süreçlerinden üretilen çıktılarla beslenen bir üst katman olarak ele alır. Bu nedenle 12207 süreçlerinden üretilen dokümanlar, test sonuçları ve kayıtlar, assurance case için sonradan toplanan ek materyaller değil; projenin doğal çıktıları olarak değerlendirilir.

Bu entegrasyonun en önemli avantajı, güvence faaliyetlerinin proje sonunda "geriye dönük" olarak kurgulanması ihtiyacını ortadan kaldırmasıdır. ISO/IEC 12207 süreçleri planlı biçimde işletildiğinde; gereksinimlerin tanımlanması, tasarım kararlarının belgelenmesi, doğrulama ve geçerleme faaliyetlerinin kayıt altına alınması ve değişikliklerin kontrol edilmesi zaten assurance case'in ihtiyaç duyduğu kanıt altyapısını üretir. ISO/IEC 15026 yaklaşımı ise bu kanıtları, üst seviye iddialarla ilişkilendirerek anlamlı bir bütün hâline getirir.

Pratikte bir projede ISO/IEC 15026 yaklaşımı uygulanacaksa—özellikle güvenlik, emniyet veya bütünlük açısından yüksek kritiklik söz konusuysa—proje yönetim planına ek olarak bazı başlıkların açık biçimde tanımlanması gerekir. Öncelikle hangi üst seviye iddiaların hedeflendiği netleştirilmelidir; örneğin "sistem yetkisiz erişime karşı korunmaktadır" veya "kritik veriler bütünlüğünü kaybetmeden işlenmektedir" gibi. Bu iddialar belirlendikten sonra, ISO/IEC 15026-3 kapsamında hangi öğelerin hangi bütünlük seviyesine atanacağı ve bu atamanın gerekçesi ortaya konur.

Buna paralel olarak, hangi tür kanıtların kabul edilebilir olduğu tanımlanır. Test raporları, statik analiz çıktıları, bağımsız inceleme kayıtları veya operasyonel ölçümler gibi kanıt türlerinin hangilerinin hangi iddiaları destekleyeceği önceden belirlenmediğinde, assurance case parçalı ve tutarsız bir yapı kazanabilir. Ayrıca bu kanıtların kimler tarafından, hangi aşamada ve hangi sürüm için üretileceği de açıkça planlanmalıdır. Son olarak, assurance case'in dayandığı varsayımlar ve dış bağımlılıkların nasıl yönetileceği tanımlanarak, güvence argümantasyonunun sınırları şeffaf hâle getirilir.

Sonuç olarak ISO/IEC 12207 ile ISO/IEC 15026'nın birlikte kullanımı, yazılım projelerinde süreç disiplinini ve güvenceyi birbirinden kopuk iki alan olmaktan çıkarır. 12207, yaşam döngüsünü düzenleyen sağlam bir iskelet sunarken; 15026 bu iskelet üzerinde, üretilen çıktılardan güven tesis eden bir üst katman inşa eder. Bu bütünleşik yaklaşım, projelerin yalnızca "doğru şekilde yürütüldüğünü" değil; aynı zamanda ortaya çıkan ürünün neden güvenilir kabul edilebileceğini de kanıta dayalı biçimde ortaya koyar.

7.ISO Standartlarının Karşılaştırmalı Değerlendirmesi

Yazılım mühendisliği alanında kullanılan ISO ve benzeri standartlar, her ne kadar ortak bir kalite ve güvenilirlik hedefini paylaşsalar da, bu hedefe farklı bakış açıları ve odak noktaları üzerinden yaklaşmaktadır. Bu nedenle ISO/IEC 12207 ve ISO/IEC 15026'nın, diğer yaygın

standartlarla birlikte değerlendirilmesi; bu standartların birbirlerinin alternatifi mi yoksa tamamlayıcısı mı olduğunun anlaşılması açısından önemlidir.

ISO/IEC 12207, yazılım yaşam döngüsünü süreç temelli bir yaklaşımla ele alır ve esas olarak "hangi faaliyetler, hangi sorumluluklar altında ve hangi çıktılarla yürütülmelidir" sorusuna yanıt verir. Bu yönüyle standart, yazılım geliştirme ve yönetim faaliyetlerinin tanımlanması ve yapılandırılması üzerine odaklanır. ISO/IEC 12207, süreçlerin nasıl uygulanacağına dair ayrıntılı yöntemler dayatmaz; bunun yerine, yaşam döngüsü boyunca gerekli görülen süreç, aktivite ve görev kümelerini tanımlayarak organizasyonlara uyarlama esnekliği sağlar. Dolayısıyla bu standart, süreç disiplini kurmak isteyen organizasyonlar için temel bir referans niteliğindedir.

Buna karşılık CMMI (Capability Maturity Model Integration), süreçlerin tanımlanmasından ziyade, bu süreçlerin olgunluk seviyesini ölçmeye ve geliştirmeye odaklanır. CMMI yaklaşımında temel soru "hangi süreçler var?" değil; "bu süreçler ne kadar tutarlı, ölçülebilir ve iyileştirilebilir biçimde uygulanıyor?" şeklindedir. Bu bağlamda ISO/IEC 12207, CMMI için bir "altyapı" olarak değerlendirilebilir. Bir organizasyon, 12207 ile süreçlerini tanımlayıp yapılandırdıktan sonra, CMMI modeli ile bu süreçlerin olgunluk seviyesini değerlendirebilir ve iyileştirme yol haritası oluşturabilir. Bu durum, iki yaklaşımın rakip değil; birbirini tamamlayan çerçeveler olduğunu göstermektedir.

Emniyet kritik alanlarda kullanılan IEC 61508 veya ISO 26262 gibi standartlar ise, ISO/IEC 12207 ve CMMI'dan farklı olarak, doğrudan emniyet hedefini merkeze alır. Bu standartlar, özellikle donanım–yazılım bütünlüğü içinde çalışan sistemlerde, belirli emniyet seviyelerinin sağlanmasını zorunlu kılar. Emniyet gereksinimleri, doğrulama kapsamı ve bağımsız değerlendirme düzeyi bu seviyelere göre önceden tanımlanmıştır. Dolayısıyla bu standartlar, "hangi güvenlik/emniyet seviyesine ulaşılması gerektiğini" normatif biçimde dayatır ve bu seviyelere ulaşmak için asgari gereksinimleri belirler.

ISO/IEC 15026 ise bu standartlardan belirgin biçimde ayrılır. 15026, doğrudan bir güvenlik veya emniyet seviyesi dayatmaz; bunun yerine bir sistemin veya yazılımın belirli bir özelliği sağladığına dair neden güvenilebileceğini açıklayan, iddia–argüman–kanıt temelli bir güvence yaklaşımı sunar. Bu yönüyle ISO/IEC 15026, "sonuç" odaklı değil; gerekçelendirme ve ikna edicilik odaklı bir standarttır. Bir sistemin güvenli, emniyetli veya güvenilir olduğunu varsaymak yerine, bu iddianın hangi kanıtlara ve hangi mantık zincirine dayandığı açıkça ortaya konur.

Bu özellik, ISO/IEC 15026'yı diğer standartlar için güçlü bir tamamlayıcı çerçeve hâline getirir. Örneğin IEC 61508 veya ISO 26262 kapsamında belirlenmiş bir emniyet seviyesinin sağlandığı iddia ediliyorsa, ISO/IEC 15026 yaklaşımı bu iddianın nasıl kanıtlandığını yapılandırmak için kullanılabilir. Benzer şekilde ISO/IEC 12207 kapsamında yürütülen doğrulama, geçerleme, denetim ve konfigürasyon yönetimi süreçlerinden üretilen iş ürünleri; ISO/IEC 15026 assurance case içinde sistematik kanıtlar olarak konumlandırılabilir. Bu durum, farklı standartlar arasında dikey entegrasyon kurulmasını mümkün kılar.

Karşılaştırmalı olarak değerlendirildiğinde; ISO/IEC 12207 “ne yapılmalı”, CMMI “ne kadar olgun yapılıyor”, emniyet standartları “hangi seviye sağlanmalı” sorularına yanıt verirken; ISO/IEC 15026 “neden güvenmeliyiz” sorusunu merkezine alır. Bu nedenle ISO/IEC 15026’nın gücü, tek başına tüm kalite veya güvenlik gereksinimlerini çözmesinde değil; diğer standartların ürettiği çıktıları anlamlı ve ikna edici bir güvence argümantasyonu içinde birleştirebilmesinde yatmaktadır.

Sonuç olarak ISO/IEC 12207 ve ISO/IEC 15026, yazılım mühendisliği ekosisteminde birbirini dışlayan değil, farklı boyutları tamamlayan standartlardır. ISO/IEC 12207 süreç disiplinini ve yaşam döngüsü düzenini kurarken; ISO/IEC 15026 bu düzen içinde üretilen çıktılardan güvence üretilmesini sağlar. Bu bütüncül bakış açısı, özellikle yüksek kritiklik gerektiren yazılım sistemlerinde kalite, güvenlik ve güvenilirliğin yalnızca sağlanmasını değil, kanıta dayalı biçimde gösterilmesini mümkün kılar.

8. Sonuç

Bu çalışma kapsamında ISO/IEC 12207:1995 ve ISO/IEC 15026 standartları, yazılım mühendisliği bağlamında süreç disiplini ve kanıta dayalı güvence perspektifleri üzerinden ele alınmış; her iki yaklaşımın tek başına sunduğu katkılar ile birlikte kullanıldıklarında ortaya çıkan bütüncül yapı ayrıntılı biçimde değerlendirilmiştir. İnceleme sonucunda, yazılım geliştirme faaliyetlerinin yalnızca teknik üretim süreçleriyle değil; planlama, kontrol, izlenebilirlik, doğrulama ve gerekçelendirme boyutlarıyla birlikte ele alınmasının, sürdürülebilir kalite ve güvenilirlik açısından kaçınılmaz olduğu görülmektedir.

ISO/IEC 12207:1995 standardı, yazılım yaşam döngüsünü rol ve süreç temelli bir yaklaşımla ele alarak, edinimden bakıma kadar yürütülecek faaliyetleri sistematik biçimde sınıflandırmaktadır. Standart; birincil, destek ve kurumsal süreç ayrımıyla, yazılım projelerinde hangi işlerin hangi sorumluluklar altında ve hangi tür çıktılarla yürütülmesi gerektiğine ilişkin ortak bir dil ve yapı sunmaktadır. ISO katalog bilgilerine göre bu sürüm “Withdrawn” statüsünde yer alsa da, süreç yaklaşımının kavramsal netliği ve yaşam döngüsü yönetimine sağladığı çerçeve nedeniyle, yazılım mühendisliği disiplinini anlamak ve öğretmek açısından hâlen temel bir referans niteliği taşımaktadır. Özellikle süreç–rol–çıktı ilişkisini görünür kılması, yazılım projelerinde yaşanan belirsizliklerin ve koordinasyon problemlerinin azaltılmasına doğrudan katkı sağlamaktadır.

ISO/IEC 15026 standart ailesi ise, yazılım ve sistemlerin belirli kritik özellikleri sağladığına dair güvenin nasıl kurulacağına odaklanarak, kalite ve güvenilirlik kavramlarını kanıta dayalı bir güvence mantığı ile ele almaktadır. Assurance case yaklaşımı (ISO/IEC 15026-2), üst seviye iddiaların sistematik argümantasyon ve kanıtlarla gerekçelendirilmesini mümkün kılmakta; integrity level yaklaşımı (ISO/IEC 15026-3) ise, sistem bileşenlerinin kritiklik düzeyine göre güvence gereksinimlerinin farklılaştırılmasını sağlamaktadır. ISO/IEC/IEEE 15026-4 ile güvence faaliyetlerinin yaşam döngüsü süreçleriyle ilişkilendirilmesi, assurance yaklaşımının statik bir doküman olmaktan çıkıp, ürünle birlikte evrilen bir yönetim nesnesi hâline gelmesini desteklemektedir.

Bu iki standart birlikte değerlendirildiğinde, aralarındaki tamamlayıcı ilişki açık biçimde ortaya çıkmaktadır. ISO/IEC 12207, yazılım yaşam döngüsünde hangi süreçlerin yürütüleceğini ve hangi çıktıların üretileceğini tanımlarken; ISO/IEC 15026, bu çıktılardan hangi iddialar için nasıl kanıt üretileceğini yapılandırmaktadır. Başka bir ifadeyle 12207, yazılım geliştirme faaliyetlerini disiplin altına alırken; 15026, bu disiplin içinde üretilen sonuçların neden güvenilebilir olduğunu gerekçelendirmektedir. Bu entegrasyon sayesinde proje, yalnızca süreçlere "uyuyor" olmakla kalmaz; aynı zamanda kalite, güvenlik veya güvenilirlik iddialarını savunabilir ve denetlenebilir biçimde ortaya koyar.

Sonuç olarak, ISO/IEC 12207 ve ISO/IEC 15026'nın birlikte kullanımı, yazılım projelerinde iki temel ihtiyaca aynı anda yanıt vermektedir: süreçlerin tutarlı ve izlenebilir biçimde yürütülmesi ve çıktıların kanıta dayalı olarak gerekçelendirilmesi. Bu yaklaşım, özellikle hata maliyetinin yüksek olduğu, regülasyon baskısının bulunduğu veya kritik niteliklerin ön planda olduğu sistemlerde, yazılım mühendisliği uygulamalarını daha şeffaf, denetlenebilir ve güvenilir hâle getirmektedir. Dolayısıyla bu iki standardın birlikte ele alınması, yazılım yaşam döngüsünün yalnızca "nasıl yönetildiğini" değil; aynı zamanda "neden güvenilebilir olduğunu" da ortaya koyan bütüncül bir kalite ve güvence perspektifi sunmaktadır.

9.Kaynakça

1. ISO/IEC 12207:1995 — ISO Standard sayfası (Withdrawn; abstract ve sürüm bilgisi).
2. ISO/IEC 12207:2008 — ISO Standard sayfası (Withdrawn; yeni sürüm bilgisi).
3. ISO/IEC 15026-2:2011 — ISO Standard sayfası (Assurance case tanımı).
4. ISO/IEC 15026-3:2011 — ISO Standard sayfası (Integrity levels kapsamı).
5. ISO/IEC/IEEE 15026-4:2021 — ISO Standard sayfası (Yaşam döngüsü üzerine süreç görünümü).
6. IEEE Standards — ISO/IEC/IEEE 15026-4 açıklaması (12207/15288 üzerine konum).
7. ISO/IEC/IEEE 15026-1 (Vocabulary & Concepts) — ISO/IEEE sayfaları (kavramsal çerçeve).
8. ISO/IEC 12207 süreç listesi .
9. https://www.geeksforgeeks.org/system-design/system-design-life-cycle-phases-models-and-use-cases/