SPL
Sermaye Piyasası
Lisanslama Sicil ve
Eğitim Kuruluşu

Lisanslama Sınavları
Çalışma Notları

Bilgi Sistemleri Geliştirilmesi ve Uygulanması

102

Hazırlayanlar:
Cem ERGÜL, Sakine YÜKSEL, Nezihe UYGUN


SPL
Sermaye Piyasası
Lisanslama Sicil ve
Eğitim Kuruluşu

Sermaye Piyasası Lisanslama Sicil ve Eğitim Kuruluşu

Bilgi Sistemleri Geliştirilmesi ve Uygulanması

Ders Kodu: 1021

- Bilgi Sistemleri Bağımsız Denetim Sınavı

31 Aralık 2024

Bu çalışma notu Cem ERGÜL, Sakine YÜKSEL, Nezihe UYGUN tarafından hazırlanmıştır.

SINAV ALT KONU BAŞLIKLARI
BİLGİ SİSTEMLERİ GELİŞTİRİLMESİ VE UYGULANMASI

1. Proje Yönetimi
- **1.1.** Proje Planlaması
- **1.2.** Proje İşletimi

2. Sistem Geliştirme Yaşam Döngüsü
- **2.1.** Sistem Geliştirme Süreçleri
- **2.2.** Sistem Bakım ve Destek Süreçleri
- **2.3.** Uygulama Kontrolleri

İÇİNDEKİLER

KISALTMALAR

BPR : Business Process Reengineering (İş Süreçlerinin Yeniden Yapılandırılması)
BT : Bilgi/Bilişim Teknolojileri
CASE : Computer Assisted Software Engineering (Bilgisayar Destekli Yazılım Mühendisliği)
CBSD : Component Based Software Development (Bileşen Tabanlı Yazılım Geliştirme)
CM : Configuration Management (Konfigürasyon Yönetimi)
CPM : Critical Path Method (Kritik Yol Metodu)
CSA : Cloud Security Alliance (Bulut Güvenliği İttifakı)
EVA : Earned Value Analysis (Kazanılan Değer Analizi)
GUI : Graphical User Interface (Grafiksel Kullanıcı Arayüzü)
ITIL : Information Technology Infrastructure Library (Bilgi Teknolojisi Altyapı Kütüphanesi)
OOSD : Object Oriented Software Development (Nesneye Yönelik Yazılım Geliştirme)
PERT : Project Evaluation Review Technique (Program Değerlendirme ve Kontrol Tekniği)
PMBOK : Project Management Body of Knowledge (Proje Yönetimi Bilgi Birikimi Klavuzu)
PMI : Project Management Institute (Proje Yönetim Enstitüsü)
POC : Proof of Concept (Kavram Kanıtı)
PUKÖ : Planla-Uygula-Kontrol Et-Önlem Al (PDCA: Plan-Do-Check-Act)
RAD : Rapid Application Development (Hızlı Uygulama Geliştirme)
RFP : Teklif Talebi (Request For Purchase)
SDLC : Software Develeopment Life Cycle (Yazılım Geliştirme Yaşam Döngüsü (YGYD)
SEE : Software Efford Estimation (Yazılım Çaba Tahminleri)
SMART : Specic, Measurable, Attainable, Relevant, Timebased (Belirli, Ölçülebilir, Başarılabilir/Gerçekçi, İlgili/Uygun, Süreli)
SOAP : Service Oriented Architecture Protocol (Servis Yönelimli Mimari Protokolü)
SoD : Seperation of Duties (Görevler Ayrılığı)
WDSL : Web Services Description Language (Web Servisleri Tanımlama Dili)
UAT : User Acceptance Test (Kullanıcı Kabul Testi)
UDDI : Universal Description Definition and Integration (Evrensel Tanım Tanımlama ve Entegrasyon)
YGYD : Yazılım Geliştirme Yaşam Döngüsü

Bu kitapta; proje yönetimi ve sistem geliştirme yaşam döngüsü konularına yer verilmektedir.

1. PROJE YÖNETİMİ

Proje kelimesinin kökeni, Latince "pro" ve "jectum" kelimelerinin birleşmesinden oluşur. "*Pro*" ön, ileri gibi anlamlara gelirken, "*jectum*" ise fırlatmak, çıkarmak anlamlarına gelmektedir. Projectum, ileriye doğru fırlatma, çıkarma anlamı taşımaktadır.

İşletmelerde gerçekleştirilen işlerin önemli bir kısmı proje olarak tanımlanmaktadır. Hangi işin proje olarak tanımlanacağının belirlenmesi için ilk olarak literatürdeki proje ile ilgili yapılan tanımlara bakılması gereklidir. Proje Yönetimi Enstitüsü (Project Management Institute-PMI) projeyi, "*Proje benzersiz bir ürün, hizmet ya da sonucun üretildiği tekrarlı olmayan bir çalışmadır*" şeklinde tanımlamaktadır. İngiltere Standart Enstitüsü (British Standards Institution-BSI) projeyi, birey ya da organizasyon tarafından yapılan, zaman, maliyet ve performans parametreleri kısıtında belirli bir başlangıç ve bitiş zamanı bulunan, koordineli aktiviteler topluluğu olarak tanımlamaktadır. Kalite kavramının fikir babalarından Juran projeyi, çözüm için çizelgelenmiş bir problem olarak ifade etmektedir. Bir diğer kaynakta ise proje, belirli kaynaklarla belirli bir zaman içerisinde tamamlanması gereken ve tekrarlanmayan özel faaliyetler topluluğu olarak tanımlamıştır. En genel anlamda proje, kaynakların etkin kullanımı, belirlenen takvimin takip edilmesi ve projenin sponsorunun beklentilerinin karşılanması başlıklarını içermelidir. Tüm proje tanımlarından da görüleceği üzere bir çalışmanın proje olabilmesi için aşağıda ifade edilen 4 özelliğe sahip olması gerekmektedir. Buna göre proje;

- Emsalsiz (özgün) bir ürün, hizmet ya da sonuç elde edilen,
- Tekrarlanmayan,
- Kısıtları olan (başlangıç, bitiş, bütçe, işgücü vb.),
- Planlama, yönetim ve kontrol ihtiyaçları olan ve pek çok faaliyetten oluşan

bir süreçtir.

Projeler, çeşitli alanlarda karşılaşılan ihtiyaçları karşılamak, problemleri çözmek veya yeni fırsatları değerlendirmek için vazgeçilmez araçlardır. Her projede bulunan üç ana unsurun anlaşılması çok önemlidir: kapsam, zaman ve maliyet. Bu unsurlardan birinin değişmesi, diğerlerinden birini veya her ikisini de etkiler. Kapsam unsuru, proje hedeflerini ve amaçlarını gerçekleştirmek için yürütülmesi gereken faaliyetleri belirlerken; zaman unsuru proje faaliyetlerinin tamamlanması için gereken zamanı ve maliyet de bu faaliyetlerin tamamlanması için katlanılan tutarı belirtir. Projelerin başarılı bir şekilde yürütülmesi için bu temel unsurlar kapsamında bazı temel hususlar bulunmaktadır. Bunlara aşağıda maddeler halinde yer verilmiştir:

- Proje Tanımı: Projeler, belirli bir süre ve sınırlı bir bütçe dahilinde yürütülen, belirli bir amacı veya hedefi gerçekleştirmeyi amaçlayan girişimlerdir. Proje tanımı yapılırken, projenin amacı, kapsamı, hedefleri, çıktıları ve beklenen faydaları belirtilmelidir.
- Ekip İşbirliği: Projeler, bir ekip tarafından yürütülür ve ekip üyeleri işbirliği içinde çalışır.
- Kaynak Kullanımı: Projeler, insan gücü, finansal kaynaklar ve malzemeler gibi kaynakları etkin bir şekilde kullanmayı gerektirir.
- Hedefler ve Amaçlar: Her proje, önceden belirlenmiş amaç ve hedefler doğrultusunda yürütülür ve bu hedefler projenin nedenini tanımlar. Projede ulaşılacak hedeflerin net bir şekilde belirlenmesi önemlidir. Bu hedefler, projenin amaçlarına, kapsamına ve müşteri beklentilerine uygun olmalıdır. Ayrıca, hedefler ölçülebilir ve gerçekçi olmalıdır.
- Planlama: Projeler, özgün bir planla başlar ve bu plan, projenin nasıl gerçekleştirileceği, ne zaman tamamlanacağı ve hangi kaynakların kullanılacağı gibi ayrıntıları içerir.
- Yürütme: Projenin planı uygulanmaya başlar ve ekip, belirlenen görevleri yerine getirir.
- Kontrol ve İzleme: Projeler, ilerlemelerini düzenli olarak izler ve gerektiğinde düzeltici önlemler alır.

• Risk Yönetimi: Olası riskler tanımlanır ve bu risklerin nasıl yönetileceği planlanır. Projelerde karşılaşılabilecek olası risklerin belirlenmesi, analiz edilmesi ve uygun önlemlerin alınması gerekir. Risk yönetimi, projenin beklenen sonuca ulaşmasını sağlamak için önemli bir unsurdur.

• Müşteri Memnuniyeti ve Kalite: Projeler, sonuçlarının müşteri memnuniyetini sağlayacak kalitede olmasını hedefler.

• Sonuçlandırma: Proje, hedeflere ulaşıldığında tamamlanır ve sonuçlar değerlendirilir.

Sponsor İlişkisi: Projeler, genellikle bir sponsor tarafından desteklenir ve sponsorun beklentilerini karşılamak önemlidir. Yönetim kelimesi ise Türkçe'de "*yön*" ve "*etmek*" kelimelerinin birleşiminden, yön göstermek, hedef göstermek temel anlamlarından gelmektedir. Bu anlamlar kapsamında değerlendirildiğinde, "*proje yönetimi*" ileride yapılacak bir işin yönünün ve hedeflerinin belirlenmesi, izlenmesi ve kontrolünü içermektedir.

Akademik olarak anlamı ise; sadece bir kez yapılan, belirli bir başlangıç ve bitiş tarihleri olan, hedefi ve amaç gözeten, kendine özgü işleri yer alan ve tüm bu işlerin planlanması, yürütülmesi, izlenmesi ve kontrolü ile kapatılmasını barındıran süreçler bütünüdür.

Proje yönetimi; proje kapsamındaki gereksinim ya da beklentilerin karşılanmasına yönelik ortak bir hedefe yönelmiş, örgütlenmiş etkinlikler kümesinin bilgi, beceri, araç ve tekniklerin belirlenen proje amaçları doğrultusunda zaman, maliyet ve kalite kısıtları aracılığıyla uygulanmasıdır. Projelerin başarılı bir şekilde yürütülmesi için uygun teknoloji kullanımı ve gerekli kaynakların tahsisinden başka, etkin ve başarılı bir proje yönetiminin de gerçekleştirilmesi gerekmektedir (Çimen, 1994:4). Proje yönetiminde temel amaç, tespit edilen amaçlara, sınırlı kaynaklarla, belli bir zaman içinde ve belli bir bütçeyle optimum şekilde ulaşmaktır.

İşletmelerde gerçekleştirilerek bilgi sistemleri denetimlerinde, bilgi sistemleri denetçisi proje yönetimi kapsamında işletme tarafından kullanılan standart veya yapıya aşina olmalıdır. Proje yönetimi yaşam döngüsü, bir projenin başlangıcından tamamlanmasına kadar tüm aşamaları içeren birleşik bir modelidir. Bu aşamalar, bağımlılıklar ve bireysel sorumluluklar gibi daha küçük ölçekli (ancak yine de önemli) unsurlardan, zamanlama, bütçe gibi önemli kilometre taşlarıyla bir projenin temel yapısına kadar her yönünü detaylandırır. Proje yönetim süreçleri aşağıdakileri aşamaları içerir:

1. Başlatma (Initiation):

Başlatma aşaması, bir proje fikrinin oluşturulduğu ve başlatılmasına karar verildiği noktadır. Bu aşamada, projenin ne amaçla yapıldığı, temel hedefleri ve kapsamı net bir şekilde tanımlanır. Projenin gerekliliği ve finansal uygunluğu değerlendirilir. Ayrıca, projenin başlangıcı için gereken kaynaklar ve ekip üyeleri belirlenir.

Projelerde başlatma adımı aşağıdaki safhalardan oluşur:

• Proje Başlangıcı: Proje başlatma aşaması, projenin neden başladığını ve işletme için ne kadar önemli olduğunu açıklar. Projeyi başlatmanın gerekliliği ve işletmeye sağlayacağı faydalar net bir şekilde belirlenir.

• Paydaşlar: Bu aşamada, projeden etkilenecek veya projeye katkı sağlayacak olan paydaşlar tanımlanır. Proje sponsorları ve kilit paydaşlar bu aşamada belirlenir.

• Proje Amaçları ve Hedefler: Projede neyin başarılması gerektiği net bir şekilde tanımlanır. Proje amaçları ve ölçülebilir hedefler belirlenir.

• Proje Yönetimi Ekip ve Görevleri: Proje yönetim ekibi oluşturulur ve her bir üyenin sorumlulukları ve görevleri belirlenir.

2. Planlama (Planning):

Planlama aşaması, projenin ayrıntılı bir yol haritasının oluşturulduğu aşamadır. Projenin kapsamı, hedefleri, bütçesi, zaman çizelgesi, riskleri ve kaynakları ile grup ve bireysel sorumluluklar, bu aşamada planlanır. Proje planı, projenin nasıl yürütüleceği konusunda rehberlik eder. İş akış planları, görev atamaları ve projenin yönetim yapısı bu aşamada hazırlanır.

Projelerde planlama adımı aşağıdaki safhalardan oluşur:

• Kapsam Tanımlama: Proje kapsamı, nelerin projeye dahil olduğu ve nelerin dışarıda bırakıldığı ayrıntılı olarak tanımlanır. Kapsam belgesi oluşturulur.

• Zaman Çizelgesi Oluşturma: Projedeki aktivitelerin sıralaması ve zamanlaması belirlenir. Proje takvimi oluşturulur.

• Bütçe Planı: Proje için gereken maliyet tahmini ve kaynak dağılımı yapılır. Bütçe planı hazırlanır.

• Risk Yönetimi Planı: Projedeki potansiyel riskler tanımlanır ve bunların nasıl ele alınacağı planlanır.

• İletişim Planı: Proje paydaşları arasındaki iletişim süreçleri, iletişim araçları ve sıklığı belirlenir.

3. Yürütme (Execution):

Yürütme aşaması, projenin plana uygun bir şekilde gerçekleştirilerek hayat bulduğu dönemdir. Ekip üyeleri görevlerini yerine getirir, kaynaklar kullanılır ve projenin ilerlemesi yakından izlenir. Proje yöneticisi, projenin plana uygun bir şekilde yürütülmesini ve ortaya çıkan herhangi bir sorunun hızla çözülmesini sağlar. Bu aşamada, sürekli görev akışının bir sonucu olarak ekipler arasında ve içinde çok fazla etkileşim olur. Bu nedenle, ekiplerin verimli ve zahmetsiz iletişim için doğru araçlara sahip olduğundan ve proje yöneticilerinin bu iletişimleri yönetmek için gerekli araçlara sahip olduğundan emin olmak önemlidir.

Projelerde yürütme adımı aşağıdaki safhalardan oluşur:

• Proje İlerlemesi: Proje yürütme aşaması, projenin plana uygun olarak ilerlediği aşamadır. Proje ekibi görevlerini yerine getirir, kaynaklar tahsis edilir ve işler başlar.

• Değişiklik Yönetimi: Proje sırasında ortaya çıkan değişiklikler yönetilir. Değişikliklerin etkileri değerlendirilir ve onaylanır.

• Kaynak Yönetimi: İnsan kaynakları, malzeme ve finansal kaynaklar etkin bir şekilde kullanılır.

• İlerleme İzleme: Proje ilerlemesi düzenli olarak izlenir ve raporlanır. Değişiklikler veya gecikmeler belirlenir ve ele alınır.

4. Kontrol Etme (Monitoring and Controlling):

Kontrol etme aşaması, projenin ilerlemesinin sürekli olarak izlendiği ve değerlendirildiği aşamadır. Proje yöneticisi, projenin zaman çizelgesini, bütçesini ve kalitesini kontrol eder. Riskler yönetilir, performans ölçümleri yapılır ve gerektiğinde düzeltici önlemler alınır. Bu aşamada sorumluluğun çoğu proje yöneticisine düşer; ancak ekipler ve bireyler de proje sürecini kontrol edebilir ve izleyebilir. Bu aşama, projenin hedeflere ulaşmasını sağlamak için kritik bir rol oynar. Kontrol etme, yürütme kadar aktif olmasa da, yine de çok fazla katılım gerektirir. Yürütmenin yanında çalıştığı için, bunu ayrı bir aşama olarak görmek; ancak iki aşamayı dengelemekte fayda vardır. Bunun nedeni, proje yürütme planının her adımının kontrol ve yakın izleme gerektirmesidir.

Projelerde kontrol etme adımı aşağıdaki safhalardan oluşur:

• Performans Değerlendirmesi: Proje ilerlemesi, zaman çizelgesi ve bütçeye göre değerlendirilir. Proje performansı ölçülür.

• Risk Yönetimi: Proje üzerindeki riskler gözden geçirilir ve yeni risklerin tanımlanmasına yönelik süreçler izlenir.

• Kalite Kontrol: Proje sonuçlarının kalitesi düzenli olarak denetlenir ve kalite standartlarına uygunluğu sağlanır.

• İletişim Sürdürme: Proje paydaşları arasındaki iletişim süreçleri devam eder ve proje ilerlemesi düzenli olarak paylaşılır.

5. Kapanış (Closing):

Kapanış aşaması, proje tamamlandığında gerçekleşir. Projenin sonuçları kabul edilir ve işletmeye teslim edilir. Projenin performansı, başarı kriterlerine göre analiz edilir ve değerlendirilir. Tüm işler tamamlanır, kaynaklar serbest bırakılır ve projenin sonuçlarıyla ilgili bir rapor hazırlanır. Proje kapanışı, projenin resmi olarak sona erdiği noktadır.

Projelerde kapanış adımı aşağıdaki safhalardan oluşur:

• Sonuçların Sunumu: Proje sonuçları, proje sponsoru ve ilgili paydaşlara sunulur. Sonuçlar kabul edilip onaylanır.

• Proje Kapanışı: Proje resmi olarak kapatılır. Kaynaklar serbest bırakılır, ekip dağıtılır ve proje dosyaları ve dokümantasyonu düzenlenir.

• İyileştirme ve Öğrenme: Proje sonuçları ve süreçleri değerlendirilir. Başarılar ve öğrenilen dersler belgelenir ve gelecekteki projeler için kullanılır.

Bu aşamalar, projelerin başarıyla yönetilmesi ve sonuçlandırılmasını sağlamak için kritik öneme sahiptir. İşletmeler, bu aşamaları doğru bir şekilde uygulayarak projelerini etkili bir şekilde yönetebilirler.

Bilgi sistemleri denetçisinin, bilgi sistemleri denetimini gerçekleştirdiği işletmenin hedeflerinin bilgi sistemlerinin yönetim uygulamaları tarafından yerine getirildiğine dair makul güvence sağlaması için, işletmenin bilgi sistemlerini ve ilgili bileşenlerini nasıl değerlendirdiğini, geliştirdiğini, uyguladığını, idame ettirdiğini ve elden çıkardığını bilmesi büyük önem taşır. Bir projenin içeriğini analiz ederken, bilgi sistemleri denetçisinin aşağıdakiler hususları da göz önünde bulundurması gerekir:

Proje Amaçları ve Hedefleri İncelemesi:

Projede belirlenen amaçların ve hedeflerin doğruluğu kadar, bu hedeflerin işletmenin genel stratejik amaçlarıyla uyumlu olup olmadığı da büyük önem taşır. Bir bilgi sistemleri denetçisi, hedeflerin organizasyonel stratejilerle ne kadar örtüştüğünü değerlendirirken, dijitalleşme ve teknolojik altyapının desteklenip desteklenmediğini de göz önünde bulundurmalıdır. Ayrıca, hedeflerin yalnızca işletme gereksinimlerini karşılaması değil, aynı zamanda teknolojik gelişmeleri ve dijital dönüşümü nasıl beslediği de önemli bir kriterdir. Projenin başarısı, yalnızca belirlenen hedeflerin gerçekleştirilip gerçekleştirilememesiyle değil, bu hedeflerin dijital ve teknolojik evrimi nasıl yönlendirdiğiyle de ölçülmelidir. Sonuç olarak, projedeki hedefler sadece somut ve ölçülebilir olmalı değil, aynı zamanda organizasyonel gelişim ve gelecekteki teknolojik gelişmelere yönelik stratejik bir vizyon da sunmalıdır.

Proje ve Altında Yatan İş Olurluğu Arasındaki Bağlantı:

Bir proje, işletmenin genel stratejik hedeflerine ve operasyonel gereksinimlerine hizmet etmek için tasarlanmış bir araçtır, ancak bu bağlantı yalnızca hedeflerin gerçekleştirilmesiyle sınırlı değildir. Bir bilgi sistemleri denetçisi, projenin iş olurluluğunu değerlendirirken, yalnızca fizibilite analizini yapmakla kalmamalı, aynı zamanda bu projenin iş süreçleriyle uyumlu olup olmadığını da gözden geçirmelidir. İş olurluluğu, sadece kısa vadeli başarılarla değil, uzun vadeli sürdürülebilirlik, ölçeklenebilirlik ve işletmenin büyüme potansiyeliyle de ilişkilendirilmelidir. Dijital dönüşüm projeleri, özellikle bulut tabanlı çözümler ve yapay zeka uygulamaları gibi modern yazılım çözümleri, projelerin iş olurluluğunu daha kapsamlı ve verimli bir şekilde sağlamak için kritik roller üstlenebilir. Bu tür teknolojiler, projelerin sadece kısa vadeli başarıları değil, aynı zamanda uzun vadeli operasyonel verimliliği ve stratejik hedeflerle uyumu üzerinde de doğrudan etki sağlar. Bilgi sistemleri denetçisi, projelerin verimliliğini ve operasyonel stratejilerle uyumunu değerlendirirken, bu teknolojilerin entegrasyonunun, projenin başarısı üzerindeki uzun vadeli etkilerini de göz önünde bulundurmalıdır.

Proje ve Diğer Projeler Arasındaki İlişki:

Projeler arasındaki etkileşim ve koordinasyon, projelerin genel başarıları ve stratejik hedeflere ulaşmaları açısından kritik bir rol oynar. Bir bilgi sistemleri denetçisi, projeler arası ilişkileri

değerlendirirken, her projenin işletmenin stratejik hedefleri ve uzun vadeli operasyonel gereksinimleriyle nasıl uyum sağladığını analiz etmelidir. Bu süreç, yalnızca fizibilite analizlerini yapmakla kalmayıp, aynı zamanda projelerin iş süreçleriyle entegrasyon derecelerini de ortaya koyar. Projelerin iş olurluluğu, sadece kısa vadeli başarılarla değil, uzun vadeli sürdürülebilirlik ve ölçeklenebilirlik açısından da değerlendirilmelidir. Özellikle dijital dönüşüm projeleri, bulut tabanlı çözümler ve yapay zeka gibi modern yazılım çözümleriyle desteklendiğinde, projeler arasındaki etkileşim daha geniş kapsamlı ve etkili hale gelir. Bu bağlamda, projelerin işletme portföyündeki diğer projelerle nasıl ilişkilendirildiği ve koordinasyonun nasıl sağlandığı, uzun vadede organizasyonel hedeflere ulaşma açısından büyük önem taşır. Bilgi sistemleri denetçisi, projelerin verimliliğini ve operasyonel stratejilerle uyumunu değerlendirirken, teknolojilerin entegrasyonunun projenin başarıya ulaşmasındaki rolünü de göz önünde bulundurmalıdır.

Kapsam Değerlendirmesi:

Proje kapsamı, belirli sınırlarla tanımlandığında, hem paydaşlar hem de proje ekibi için daha net bir yön sağlar ve bu, projenin başarısını doğrudan etkileyebilir. Bilgi sistemleri denetçisi, kapsamı değerlendirirken, projeye dahil olan işlevsel alanları belirlemeli ve dışarıda bırakılması gereken unsurları dikkatle tanımlamalıdır. Kapsamın net bir şekilde tanımlanması, tüm paydaşlar arasında anlaşılabilirliği artırarak projenin yönetimini kolaylaştırır. Ayrıca, bu süreç, kapsam değişikliklerinin yönetilmesini ve yeni gereksinimlerin entegrasyonunu da kapsar. Dijital araçlar, kapsam değişikliklerinin izlenmesini kolaylaştırarak proje yönetimini daha esnek hale getirir. Modern proje yönetim yazılımları, kapsamı sürekli olarak izleyerek, proje süresince ortaya çıkabilecek değişikliklere hızla adapte olunmasına olanak tanır ve böylece projelerin daha verimli bir şekilde ilerlemesini sağlar. Kapsamın doğru bir şekilde tanımlanması ve yönetilmesi, projenin hem zaman hem de bütçe açısından başarısını garanti altına alabilir.

Bütçe ve Kaynakların İncelenmesi:

Proje bütçesi, ihtiyaçları karşılayacak şekilde dikkatlice belirlenmeli ve tahsis edilen kaynakların projenin gereksinimleriyle uyumlu olup olmadığı dikkatle değerlendirilmelidir. Bilgi sistemleri denetçisi, bütçenin doğruluğunu inceleyerek, tahsisin projenin gereksinimlerini karşılayıp karşılamadığını doğrulamalıdır. Ayrıca, kaynakların etkin bir biçimde dağıtılması, projenin verimli bir şekilde ilerlemesi için kritik bir rol oynar. Bu bağlamda, dijital projelerde bulut bilişim gibi esnek çözümler ve yapay zeka destekli kaynak yönetimi araçları kullanılarak, bütçenin daha etkili bir şekilde kontrol edilmesi sağlanabilir. Modern proje yönetim yazılımları, otomatik araçlar aracılığıyla kaynakları daha verimli bir şekilde planlamayı ve bütçeyi daha doğru tahmin etmeyi mümkün kılar. Kaynakların doğru bir şekilde tahsis edilmesi, projenin zamanında tamamlanmasını ve hedeflenen sonuçların elde edilmesini sağlar. Böylece, kaynakların verimli kullanımı ve bütçenin doğru yönetilmesi, projenin başarıyla tamamlanmasında önemli bir etken olur.

Zaman Çizelgesinin İncelenmesi:

Bilgi sistemleri denetçisi, proje zaman çizelgesini değerlendirirken, belirlenen hedeflere ulaşmayı ve teslim tarihlerini karşılamayı sağlayacak stratejik bir yaklaşım sergilemelidir. Zaman çizelgesindeki değişikliklerin, projenin genel ilerleyişi ve hedeflere ulaşılmasındaki etkisi de göz önünde bulundurulmalıdır. Zaman çizelgesinin dinamik olması, değişen koşullara hızla adapte olabilmesi önemlidir, çünkü her projede öngörülemeyen faktörler ortaya çıkabilir. Proje yönetimi yazılımları, zaman çizelgesi takibi ve varyans analizi gibi özellikleri ile projelerin düzgün ilerlemesini sağlar ve olası sapmaları erken aşamada tespit eder. Bilgi sistemleri denetçisi, bu araçları kullanarak sapmaları ve olası gecikmeleri hızlı bir şekilde belirler, böylece zamanında tamamlanma için gerekli önlemleri alır. Ayrıca, zaman çizelgesindeki gecikmelerin veya değişikliklerin projenin sonuçları üzerindeki etkisini değerlendirmek, projedeki riskleri yönetmek ve hedeflere ulaşma konusunda stratejik kararlar almak için kritik bir aşamadır. Bu değerlendirme, projelerin başarılı bir şekilde tamamlanmasını sağlamak için önemli bir rol oynar.

Risk Analizi ve Yönetimi:

Proje başarısının sağlanabilmesi için, karşılaşılabilecek olası tehditlerin ve engellerin önceden belirlenmesi büyük önem taşır. Bilgi sistemleri denetçisi, bu potansiyel risklerin projenin ilerleyişine

olan etkilerini değerlendirirken, risk yönetimi planlarının uygulanabilirliğini de gözden geçirmelidir. Risk yönetimi, sadece olası sorunları tanımlamakla kalmaz, aynı zamanda bu sorunların etkilerini en aza indirmek için alınacak önlemleri de belirler. Günümüzde, yapay zeka destekli analizler ve ileri düzey veri analizi gibi modern teknikler kullanılarak, risklerin daha doğru ve zamanında tespit edilmesi sağlanabilir. Bu yaklaşımlar, proje yöneticilerinin daha bilinçli kararlar almasına olanak tanır ve projenin başarısızlık riskini minimize eder. Risklerin etkili bir şekilde yönetilmesi, proje sürecinde esneklik sağlar ve beklenmedik durumlarla başa çıkmayı kolaylaştırır. Bu süreç, projenin hedeflerine ulaşması için gerekli adımların zamanında atılmasını sağlar ve projedeki belirsizlikleri minimize eder.

İş Sürekliliği ve Kriz Yönetimi Planlarının İncelenmesi:

Proje planlamasında, beklenmedik durumlarla başa çıkabilme yeteneği büyük önem taşır ve iş sürekliliği ile kriz yönetimi stratejilerinin etkinliği bu noktada belirleyici rol oynar. Bilgi sistemleri denetçisi, iş sürekliliği ve kriz yönetimi planlarının etkinliğini değerlendirirken, bu stratejilerin projedeki kritik süreçleri ne kadar güvence altına aldığını incelemelidir. Bu süreç, projenin herhangi bir aksaklık durumunda hızla toparlanmasını sağlamak ve operasyonel sürekliliği korumak için kritik bir öneme sahiptir. Dijital projelerde, modern teknolojilerle desteklenen veri yedekleme ve bulut tabanlı çözümler, iş sürekliliğini sağlamada önemli bir rol oynar. Kriz yönetimi planlarının düzenli olarak test edilmesi ve tüm paydaşlarla paylaşılması, projenin olası aksaklıklara karşı dayanıklılığını artırır ve gerektiğinde hızlı bir şekilde tepki verilmesine olanak tanır. Bu yaklaşımlar, projenin beklenmedik durumlarla başa çıkabilme yeteneğini güçlendirir ve kriz durumlarında bile hedeflere ulaşmayı mümkün kılar.

Proje Ekip ve İletişim Stratejisi İncelenmesi:

Projelerde başarı, yalnızca doğru kaynakların kullanımıyla değil, aynı zamanda doğru ekip dinamiklerinin sağlanmasıyla da doğrudan ilişkilidir. Bilgi sistemleri denetçisi, proje ekibinin yetkinliklerini değerlendirirken, her bir ekip üyesinin uzmanlık alanlarını ve projeye sağladığı katkıyı dikkatle analiz etmelidir. Ekip üyelerinin rollerinin ve sorumluluklarının net bir şekilde tanımlanması, proje başarısı için kritik bir unsurdur, çünkü bu durum, ekip içindeki işbirliği ve verimliliği artırır. Ayrıca, projede etkin bir iletişim stratejisi geliştirilmesi gereklidir; iletişim kanalları, tüm paydaşlar arasında açık ve şeffaf olmalıdır. İyi bir iletişim, proje sürecindeki yanlış anlamaları önler ve tüm paydaşların aynı hedefler doğrultusunda hareket etmesini sağlar. Dijital projelerde, iletişim süreçlerinin daha verimli hale getirilmesi için Slack, Microsoft Teams gibi platformlar kullanılabilir. Bu araçlar, proje ekibinin hızlı ve etkin bir şekilde iletişim kurmasını sağlar ve proje ilerlemesini şeffaf bir şekilde izlemeye imkan tanır, böylece tüm ekip üyeleri proje hakkında anlık bilgiye sahip olabilir. Etkili bir iletişim stratejisi, projenin her aşamasında paydaşlar arasında koordinasyonu kolaylaştırır ve proje sürecinin başarılı bir şekilde yönetilmesine yardımcı olur.

Belgeleme ve İzleme Yöntemleri İncelenmesi:

Projelerin başarısı, sadece hedeflere ulaşmakla sınırlı kalmaz, aynı zamanda bu sürecin nasıl izlenip belgelendiğiyle de yakından ilişkilidir. Bilgi sistemleri denetçisi, projenin ilerlemesini ve sonuçlarını belgelemek için kullanılan yöntemleri gözden geçirmeli ve bu süreçte ne kadar ilerleme kaydedildiğini değerlendirmelidir. Proje izleme, hedeflerin ne kadar gerçekleştirildiğini belirlemek ve herhangi bir sapmayı tespit etmek için kritik bir adımdır. Dijital projelerde, veritabanı yönetimi ve raporlama yazılımları kullanılarak proje verileri anlık olarak izlenebilir ve yönetilebilir, bu da proje yöneticilerinin hızlı ve doğru kararlar almasına olanak tanır. Proje izleme raporları düzenli olarak hazırlanmalı ve tüm kritik bilgileri içermelidir; bu raporlar, projenin ilerleyişini, karşılaşılan zorlukları ve elde edilen başarıları şeffaf bir şekilde ortaya koyar. Bu belgeler, sadece projenin başarısını ölçmek için değil, aynı zamanda gelecekteki projelerde yol gösterici bir referans kaynağı olarak da önemli bir rol oynar. Düzenli izleme ve belgeleme, projenin hedeflerine ulaşması için gerekli düzeltici önlemleri zamanında almayı mümkün kılar ve projenin başarısının sürdürülebilir olmasını sağlar.

Müşteri Memnuniyeti ve Kalite Kontrolü Değerlendirmesi:

Bilgi sistemleri denetçisi, projenin müşteri beklentilerini karşılayıp karşılamadığını ve kalite standartlarına uygun olup olmadığını analiz etmelidir. Müşteri memnuniyeti, projenin başarısının önemli bir göstergesidir ve bu nedenle müşteri geri bildirimlerinin toplanması ve değerlendirilmesi

büyük önem taşır. Modern projelerde, müşteri geri bildirimleri dijital platformlar ve anketler aracılığıyla hızla toplanarak daha etkili bir şekilde analiz edilebilir. Bu süreç, müşteri ihtiyaçlarına hızlı bir şekilde cevap verme ve proje çıktılarının sürekli iyileştirilmesi için değerli bilgiler sunar. Ayrıca, kalite kontrol süreçlerinin proje süresince sürekli izlenmesi gerekir. Kalite kontrol yazılımları ve test otomasyon araçları kullanılarak, proje çıktılarının kalitesi güvence altına alınabilir ve müşteri memnuniyeti sağlanabilir. Bu araçlar, ürünün veya hizmetin her aşamasında kaliteyi izlemeyi kolaylaştırır ve olası hataların erken aşamada tespit edilmesini sağlar. Kalite kontrol süreçlerinin etkinliği, sadece projenin başarısını artırmakla kalmaz, aynı zamanda müşteri memnuniyetini de en üst düzeye çıkarır.

Proje Sonlandırma ve Değerlendirme:

Proje sonlandırıldığında, sonuçların değerlendirilmesi ve hedeflere ulaşılıp ulaşılmadığının belirlenmesi büyük bir önem taşır. Bilgi sistemleri denetçisi, projenin başlangıçta belirlenen hedeflerle karşılaştırıldığında ne kadar başarılı olduğunu değerlendirirken, bu süreçte projedeki tüm başarılar ve eksiklikler gözden geçirilmelidir. Proje çıktılarının işletmeye nasıl aktarılacağının planlanması, projenin sürdürülebilirliğini ve uzun vadeli etkilerini sağlamak için kritik bir adımdır. Bu aktarım süreci, hem proje ekibinin hem de ilgili paydaşların doğru bir şekilde bilgilendirilmesini ve gerekli eğitimlerin verilmesini içermelidir. Projenin sonuçları, işletmeye değer yaratacak şekilde somutlaştırılmalı ve uygulanabilir hale getirilmelidir. Ayrıca, proje sonlandırma aşamasında, tüm paydaşlar tarafından yapılan geri bildirimler dikkate alınarak, gelecekteki projelerde öğrenilen derslerin nasıl uygulanacağına dair stratejik bir yol haritası oluşturulmalıdır. Bu süreç, proje sonrası değerlendirmelerin hem mevcut projenin sonuçlarını hem de gelecekteki projelerin başarısını artıracak şekilde şekillendirilmesine olanak tanır.

1.1. Proje Planlaması

Proje yönetimini yönetimden ayıran önemli bir faktör, devam eden bir süreç olan yönetimden farklı olarak, proje yönetiminin nihai, teslim edilebilir ve sınırlı bir zaman dilimine sahip olmasıdır.

Her projeye başlarken şu temel soruların cevaplandırılması gerekir:

Neden proje başlatılıyor? (Proje Rasyoneli):

- Projenin başlatılma nedeni ve amaçları net bir şekilde tanımlanmalıdır. Projeyi başlatma gerekliliği ve işletmeye sağlayacağı faydalar açıklanmalıdır.

Bu iş kimin için yapılıyor? (Paydaş Tanımlama):

- Proje kapsamında kimlerin etkileneceği ve proje sonuçlarından kimlerin faydalanacağı belirlenmelidir. Proje sponsoru ve kilit paydaşlar tanımlanmalıdır.

Ne teslim edilecek? (Proje Kapsamı):

- Projenin kapsamı ve nihai ürün, hizmet veya sonuçların açık ve ayrıntılı bir şekilde tanımlanmalıdır. Hangi işlerin tamamlanması gerektiği netleştirilmelidir.

Hangi Kaynaklara İhtiyaç Var? (Kaynak Gereksinimleri):

- Projenin başarılı bir şekilde yürütülebilmesi için fiziksel ve finansal kaynaklar belirlenmelidir. İnsan kaynakları, bütçe ve diğer gereksinimler gözden geçirilmelidir.

Ne Zaman Teslimatlar Üretilecek? (Zaman Çizelgesi):

- Projede hangi işlerin ne zaman tamamlanması gerektiği belirlenmelidir. Proje zaman çizelgesi oluşturulmalı ve takip edilmelidir.

Ne Zaman İnceleme Yapılacak? (İlerleme Kontrolü):

- Proje paydaşları, projenin ilerlemesini ne sıklıkta ve hangi aşamalarda inceleyeceklerini belirlemelidir. İlerlemenin kontrol edilmesi için zamanlamalar belirlenmelidir.

Proje Sonucu Ne Zaman Onaylanacak? (Sonuç Kabulü):

- Proje sponsoru, nihai sonuçların ne zaman kabul edilip onaylanacağını belirlemelidir. Projenin tamamlanmasının ardından sonuçların onay süreci tanımlanmalıdır.

Nerede Teslimatlar Kullanılacak? (Uygulama Alanları):

- Teslim edilecek ürünler veya sonuçlar, hangi bölgelerde veya işletme süreçlerinde kullanılacaklarına dair açıklamalar içermelidir.

Nasıl Amaçlar ve Hedeflere Ulaşılacak? (Strateji):

- Proje amaçları ve hedefleri, projenin stratejik yaklaşımını içermelidir. Gereksinimler, iş süreçleri ve yöntemler açıklanmalıdır.

Başarı Nasıl Ölçülecek? (Başarı Kriterleri):

- Projede başarıyı değerlendirmek için kullanılacak ölçütler ve metrikler belirlenmelidir. Proje sonuçlarının neye göre başarılı olduğu netleştirilmelidir.

1.1.1. Projelerde Yönetişim Yaklaşımı

Başarılı proje planlamasının genel özelliği, riske dayalı bir yönetim süreci ve doğası gereği tekrarlanabilir olmasıdır.

Aşağıda yer verilen faaliyetlerin her biri, projenin başarılı bir şekilde yönetilmesi ve kontrol edilmesi için kritik öneme sahiptir. Bilgi sistemleri denetçisi, bu süreçlerin etkin bir şekilde uygulandığını doğrulamak için değerlendirmeler yapmalıdır:

- Proje Komiteleri/Kurulu Gözetim Seviyesi: Proje komiteleri veya kurulları, projenin yönünü belirleyen ve önemli kararlar alan gruplardır. Bu komiteler, projenin hedeflerini ve stratejisini belirlerler. Denetçi, bu komitelerin etkili bir şekilde kurulup çalıştığını, projenin yönetimini desteklediğini ve gerektiğinde kararlar aldığını doğrulamalıdır.

- Risk Yönetim Yöntemleri: Risk yönetimi, projedeki potansiyel risklerin tanımlanması, analiz edilmesi ve yönetilmesini içerir. Bu faaliyet, olası sorunların önceden belirlenmesi ve projeye uygun önlemlerin alınmasını sağlar. Denetçi, risk yönetimi yöntemlerini incelemeli ve projenin risklerle nasıl başa çıktığını değerlendirmelidir.

- Sorun Yönetimi: Proje sırasında ortaya çıkan sorunların tanımlanması, sınıflandırılması ve çözülmesi önemlidir. Sorunlar hızlı bir şekilde ele alınmalı ve çözüm süreci izlenmelidir. Denetçi, sorun yönetimi süreçlerini gözden geçirmeli ve sorunların etkili bir şekilde çözülüp çözülmediğini değerlendirmelidir.

- Maliyet Yönetimi: Proje bütçesinin yönetimi ve izlenmesi kritik bir faktördür. Harcamaların kontrol edilmesi ve bütçeye uygun olarak yönetilmesi, projenin finansal başarısını etkiler. Denetçi, maliyet yönetimi süreçlerini incelemeli ve bütçenin ne kadar etkili bir şekilde yönetildiğini değerlendirmelidir.

- Planlama ve Bağımlılık Yönetimi Süreçleri: Proje planlarının oluşturulması, güncellenmesi ve takip edilmesi önemlidir. Ayrıca, farklı görevlerin ve aktivitelerin birbirine bağımlılıklarının yönetilmesi de kritik bir rol oynar. Denetçi, projenin planlama ve bağımlılık yönetimi süreçlerini incelemeli ve projenin planlara uygun ilerlediğini değerlendirmelidir.

- Raporlama Süreçleri: Projenin ilerlemesini ve performansını düzenli olarak raporlamak, paydaşlara proje hakkında güncel bilgi sağlama açısından önemlidir. Denetçi, raporlama süreçlerini değerlendirmeli ve proje ilerlemesinin doğru bir şekilde iletilip iletilmediğini kontrol etmelidir.

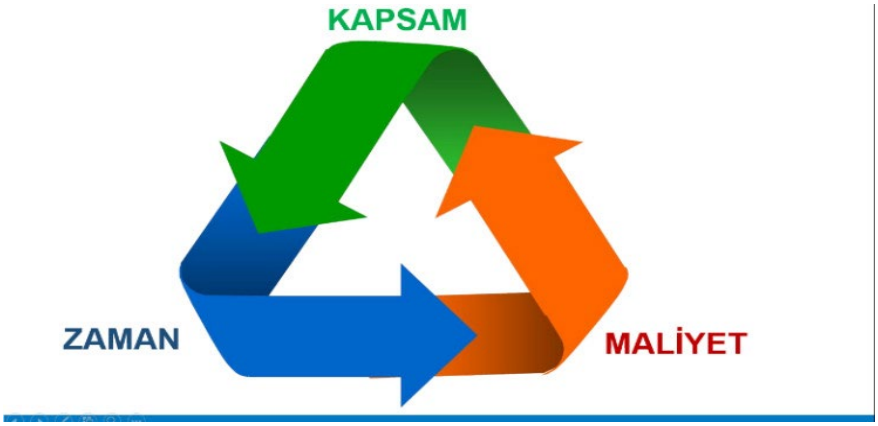
- Değişim Kontrol Süreçleri: Proje sürecinde ortaya çıkan değişikliklerin tanımlanması, değerlendirilmesi ve onaylanması süreçleri belirlenmelidir. Denetçi, değişim kontrol süreçlerini incelemeli ve değişikliklerin etkili bir şekilde yönetildiğini değerlendirmelidir.

- Paydaş Yönetiminin Katılımı: Projeye katılan tüm paydaşların ihtiyaçlarının anlaşılması ve onların aktif katılımının sağlanması, projenin başarısını etkiler. Denetçi, paydaş yönetimi süreçlerini incelemeli ve paydaşların gerektiğinde etkili bir şekilde dahil edildiğini doğrulamalıdır.

• Onay Süreci: Projede belirli aşamalarda veya dönemlerde gerekli onayların alınması ve bu süreçlerin belirli bir düzen ve prosedüre göre yürütülmesi gereklidir. Denetçi, onay süreçlerini incelemeli ve gerekli onayların uygun bir şekilde alındığını değerlendirmelidir.

1.1.1.1. Proje Yönetiminin Temel Fonksiyonları

Proje bazında bazı farklı gereklilikler ön plana çıksa da her proje faaliyet alanı, zaman ve maliyet kısıtları ile sınırlıdır. Bu sınırlara proje üçgeni veya proje saç ayakları da denir. Bunlar başarılı proje yönetimi için gereklidir.



Proje Üçgeni

Kapsam: Projenin amaçladığı hedefler sorusuna cevap arar. Hangi ürün veya hizmetin elde edileceği belirlenir.

Zaman: Projenin tamamlanması için gerekli süre sorusuna cevap aranır. Ürün veya hizmetin gerçekleştirilme takvimi belirlenir. Belirlenen süreye uyulmaması yaptırımlar gerektirebilir. Bu da projenin maliyetlerini artırır. Proje yöneticisinin görevi her adımda takvim ile gerçekleşen işi kontrol etmektedir.

Maliyet: Projenin tamamlanması durumunda katlanılacak maliyetin ne olduğu sorusuna cevap arar. Projenin tamamlanmasında kullanılacak kaynakların maliyetidir. Bu maliyetler, analist veya programcı ücretleri, yazılım, donamım ve bilgisayar ağları için yapılan harcamalar gibi projenin yürütülmesiyle doğrudan ilgili faaliyetler kapsamında katlanılan maliyetlerdir.

1.1.1.2. Proje Yönetiminin Hedefleri (3T)

Başarılı bir proje yönetimi, üç kısıt için belirlenen hedeflere ulaşılması ve müşterinin memnuniyeti ile ölçülür. Ancak proje uygulaması sırasında ortaya çıkan hedeflerden sapmalardan dolayı proje yöneticileri hileye başvurmak veya öncelik sırası oluşturmak gibi uygulamalarda bulunabilir. Üç kısıt arasında fedakârlık bulunulması durumunda;

- Kapsamı genişletmek maliyet ve zamanın artmasına neden olur.
- Zamanı azaltmak, özellikle fazla mesai, fazla kaynak gerekiyorsa, maliyetlerde artışa yol açar.
- Maliyetleri düşürmeye çalışmak proje takvimini olumsuz yönde etkileyebilir. Hatta kapsamdan ödün verilmesine neden olabilir.
- Kapsamı düşürmek her ne kadar maliyet ve zaman kısıtında azalış sağlasa da kalite kavramını olumsuz yönde etkileyebileceğinden müşterinin tatminsizliğine yol açar. Bu durumda, yeniden düzenleme, zamanda gecikme, verimlilikte düşüş ve maliyetlerin artmasına neden olur.
- Devam eden bir projeye ilave kaynak eklemek (insan veya fiziksel kaynak) proje bütünlüğüne zarar verebileceğinden verimsizliğe yol açabilir.
- Bitiş sürelerini ötelemek bilgi sistemleri projelerinin insan (analist, programcı) odaklı projeler olması nedeniyle maliyetleri azaltmayacaktır. Zaman para demektir.

Dolayısıyla kısıtların birinde yapılacak değişiklik diğerlerini de etkilemektedir. Üç kısıt proje yönetiminin temel faktörlerini temsil eder. Ancak bunların dışında kalite de önemli bir faktördür. Her üç kısıtı da sağlayan bir proje, kalite açısından yetersiz ise müşteri tarafından kabul edilmeyecektir. Bu nedenle müşteri memnuniyeti açısından kalite de önemli bir faktördür.

Sonuçta, projenin her yönetim faaliyeti bu faktörlerden sadece birini değil, tümü düşünülerek kararlaştırılmalıdır. Proje yönetimi, kapsam, zaman ve maliyet kısıtlarının her üçünü kalite çemberi içerisinde yönetebiliyorsa, iyi bir proje yönetimi için doğru yoldadır.

1.1.1.3. Proje Yönetim Uygulamaları

Literatürde bazı çevreler tarafından Mısır Piramitleri'nin yapımı veya Çin Seddi'nin inşaatı bir proje yönetimi olarak kabul edilse de proje yönetiminin modern konseptinin atom bombasının geliştirildiği Manhattan Projesi ile başladığını kabul edilmektedir. Modern proje yönetimi teknikleri, 19. yüzyılın sonlarında artan ve karmaşıklaşan iş yaşamı ile birlikte şekillenen ve gelişen yönetim ilkelerinin dönüşümü ile oluşmuştur. Özellikle büyük ölçekli kamu projeleri, anılan tekniklerin gelişmesinde itici güç olmuştur. 20. yüzyılın başlarında Frederick Taylor'un yönetim tekniklerinin bilimsel olarak analiz edilebileceğini ve geliştirilebileceğini keşfetmesi ile birlikte yönetim anlayışında yeni bir sayfa açılmıştır. Taylor'un çalışmalarından önce verimliliği artırmanın tek yolu işçilerin daha uzun saatler boyunca daha sıkı çalıştırılmasıydı. Taylor iş süreçlerini, en basit parçalarını tek tek analiz ederek, daha verimli hâle getirmiştir (Kazan, 14).

Proje yaşam döngüsü boyunca kullanılan birçok proje yönetim tekniği ve araçları bulunmaktadır. Söz konusu teknik/araçlar projenin boyutuna, karmaşıklığına göre manuel veya otomatize sistemler olabilmektedir. Bahsedilen teknikler yazılım boyutu tahmini, zamanlama, kaynak tahsisi ve üretkenlik vb. hususlarda ölçüme dayalı nitel/nicel yaklaşımlar sağlamaktadır.

Proje yönetiminin çıktı, zaman ve bütçe olmak üzere üç temel kısıtı önem arz etmektedir. Projenin bütçe ve zaman kısıtları çıktıların özellikleri ile ilişkili olmalıdır. Genelde beklentisi yüksek olan çıktılar, uzun süre ve yüksek bütçe arasında pozitif bir korelasyon oluşturmaktadır. Projenin ihtiyaç duyduğu kaynaklar belirli tahmin teknikleri ile projenin başlangıcında tahminlenir. Bu kapsamda gerçekleştirilen boyut tahmini kaynakların toplamına ilişkin bir hesaplama sağlar.

Proje yönetimi, bir projenin planlanması, izlenmesi, kontrol edilmesi ve başarıyla tamamlanması için kritik bir rol oynar. Bu süreçler, projenin bütçe, zaman ve çıktı gibi temel kısıtlarıyla yakından ilişkilidir.

Proje yönetimi süreçlerinde kullanılan teknikler ve araçlar ve bunların etkin olarak proje süresi boyunca kullanılması projenin başarısında kritik rol oynamaktadır. Aşağıda maddeler halinde projenin başarısına etki eden teknik ve araçlar hakkında bilgilere yer verilmiştir:

1. Proje Yönetim Yazılımları: Proje yönetim yazılımları, projenin planlanması, kaynak tahsisi ve ilerlemesinin izlenmesi için önemli bir rol oynar. Bu yazılımlar, projenin zaman çizelgesini oluşturmanıza, kaynakları tahsis etmenize ve proje ilerlemesini gözden geçirmenize olanak tanır. Microsoft Project, Primavera P6, Asana ve Trello gibi yazılımlar, projelerin yönetilmesine yardımcı olur.

2. Zaman Çizelgesi ve Proje Takip Araçları: Projelerin zaman çizelgeleri, projenin hangi aşamada olduğunu ve ne zaman tamamlanacağını anlamak için kullanılır. Bu çizelgeler, proje yöneticilerinin işleri planlamasına, kaynakları tahsis etmesine ve takip etmesine yardımcı olur. Gantt şeması gibi araçlar, projenin zaman çizelgesini görsel olarak temsil eder.

3. Kaynak Yönetimi Araçları: Projeler, belirli kaynakların tahsis edilmesini gerektirir. Bu kaynaklar insanlar, bütçe, malzeme ve teknoloji gibi çeşitli unsurları içerebilir. Kaynak yönetimi araçları, bu kaynakların planlanmasını ve izlenmesini kolaylaştırır. Projeye hangi kaynakların ne kadar süreyle tahsis edileceği, proje başlangıcında tahmin edilir.

4. Maliyet Tahmini Araçları: Proje bütçesini ve maliyetlerini tahmin etmek, projenin başarısı için kritiktir. Maliyet tahmini araçları, proje maliyetlerini analiz etmeye ve izlemeye yardımcı olur. Bu araçlar, proje harcamalarını kontrol etmeye ve projenin bütçeyle uyumlu olmasını sağlamaya yardımcı olur.

5. Kalite Kontrol ve Kalite Güvence Araçları: Projelerin başarılı olması sadece zaman ve maliyetle değil, aynı zamanda kaliteyle de ilgilidir. Kalite kontrol ve kalite güvence araçları, projenin kalitesini yönetmek için kullanılır. Bu araçlar, standartlar, yönergeler ve denetimler aracılığıyla kaliteyi sağlama konusunda yardımcı olur.

Tüm bu teknikler ve araçlar, projenin başarısını artırmak, riskleri azaltmak ve çıktıların kalitesini artırmak için projenin farklı aşamalarında kullanılır. Proje yönetimi, projenin hedeflerine ulaşmasına yardımcı olurken aynı zamanda bütçe ve zaman kısıtlamalarını da yönetir.

a. Proje Portföyü ve Program Yönetim Kavramları

Proje portföyü, bir işletmenin finansal ve stratejik amaçlarını gerçekleştirmek için birlikte yönetilen ve optimize edilen proje, program ve süreçler topluğu olarak adlandırılabilir. Proje portföy yönetimi, bir grup projenin bir bütün olarak ele alınıp önceliklendirilmesi, yürütülmesi ve getirilerinin izlenmesi olarak da düşünülebilir. Proje portföyü, bir işletmenin tüm projelerini kapsayabileceği gibi, işletme içerisindeki bir birimin portföylerini de kapsayabilir. Örneğin bir işletmenin üretim bölümü ile bilgi işlem bölümünün ayrı proje portföyleri olabilir.

Proje portföy yönetimi, işletmenin genel stratejisiyle uyum içerisinde, portföy dengelerini gözeterek ve projenin değerini maksimize ederek dahili ve harici kısıtlara tabi olan bir girişimin başarısını ve genel zenginliğini artırmak amacıyla organizasyona ait projelerin yönetimi olarak da açıklanır (Moustafaev, 2017: 5). Temel odak noktası, portföydeki projelerin sonuçlarının işletmenin sratejik hedeflerini desteklemektir.

Proje portföy yönetimi, esasında projenin yaşam döngüsünün çok ötesinde bir iş olup, bir yandan olası bir yatırım projesinin işletmenin faaliyet sahası arasındaki geleneksel boşlukları doldurmaya çalışırken öte yandan sınırlı kaynaklardan maksimum değeri sunma arasındaki bağı kurmaya çalışmaktadır (Levine, 2005: 13).

Proje portföy yönetiminde, klasik bir proje yönetiminin dikkate aldığı gibi projelerin ne zaman bitirileceği veya ne kadara mal olacağının dışında aşağıdaki sorulara cevap aranmaktadır (Levine, 2005: 16):

- Ne gibi potansiyel projelerin birleşimi, işletme için uzun vadede büyümeyi sağlamak ve yatırım getirisini maksimize etmek için insan ve nakit kaynaklarının en iyi şekilde kullanılmasını sağlar?

- Bu projeler işletmenin stratejik girişimlerini nasıl destekler?

- Bu projeler işletme paylarının değerini nasıl etkiler?

Bir proje portföyü aşağıdaki özelliklere sahip olabilir:

- Çeşitlilik: Proje portföyü, farklı türde projeleri içerebilir. Örneğin IT projeleri, inşaat projeleri, pazarlama projeleri vb.

- Öncelikler: Projeler, organizasyonun stratejik önceliklerine göre sıralanır ve yönetilir.

- Kaynaklar: Proje portföyü yönetimi, organizasyonun kaynaklarını projeler arasında dengeli bir şekilde dağıtmasına yardımcı olur. Bu, bütçe, insan kaynakları ve diğer kaynakları içerir.

- Riskler ve Getiri: Her proje, riskleri ve beklenen getiriyi içerir. Proje portföyü yönetimi, organizasyonun riskleri dengelemesine ve projelerin sağlayabileceği getiriyi değerlendirmesine yardımcı olur.

- İzleme ve Değerlendirme: Proje portföyü yönetimi, projelerin ilerlemesini izler, performansı değerlendirir ve gerektiğinde ayarlamalar yapar.

Proje portföy yönetiminin sorguladığı hususlar dikkate alındığında, proje yönetiminden farklı olarak, doğası itibariyle kötü bir projenin mükemmel yönetilerek belli bir düzeye getirilmesinden çok, hangi projenin işletme tarafından yüklenilmesi gerektiğine odaklanılmaktadır. Proje portföy yönetimi, hali hazırda yürütülmekte olan projeler ile entegrasyon sağlamanın önemini vurgulamasının yanı sıra, potansiyel projelerin işletmenin gelecekteki kârlılığına nasıl bir etkide bulunacağı temelinde düşünmeye neden olması bakımından da önemlidir ve olası projelerin nasıl seçilip yönetileceğine odaklanmaktadır (Levine, 2005: 17).

Proje portföy yönetimi, temel olarak iki bölüme ayrılabilir. Bunlardan ilki önceliklendirme ve portföye alınacak yatırım projesinin seçimi, ikincisi ise portföye alınan projelerin yönetimi (Levine, 2005: 23). Potansiyel yatırım projelerinin seçimine geçmeden önce portföyün sınırlarını belirlemek için

her bir portföy doğası itibariyle bileşenleriyle birlikte karşılıklı ilişkiye sahip olduklarından proje portföy yöneticilerinin aşağıdaki eylemleri gerçekleştirmeleri gerekir (Chatzipanos, 2018: 20; Levine, 2005: 24):

1. Portföyün kendi içindeki fizibilite, açık teslim süreleri, yarattığı katma değer, sınırlılıklar gibi bileşenlerini kavramalı,

2. İşletmenin stratejik hedefleri de dahil olmak üzere bu bileşenler arasındaki ilişkileri kavramalı,

3. Portföyün çevre şartlarını anlamalı,

4. Tüm portföy ekosistemini kavramalı,

5. Portföy içinde projelerin birbiriyle olan sinerji durumlarını analiz etmeli,

6. Stratejik değişim sürecini de takip ederek portföyü ekosistem değişimlerine hazırlamalı,

7. Değerlendirme ve önceliklendirme yapmalı ve ilgili değişimlerin varlığını kanıtlamalı,

a. Değerlendirme ve önceliklendirmelerde projenin değer ile fayda takdirini yapmalı ve bunlara ilişkin risk analizi gerçekleştirmeli,

b. Değerlendirme ve önceliklendirme için en uygun kaynak kullanımını belirlemeli,

c. Seçim kriterleri setine uygun olarak projeleri sıralamalı.

8. Performanslarını da öngörerek söz konusu değişimleri hayata geçirmeli.

Projelerin seçiminden sonra iki boyutta proje portföyünü izlemeye almak gerekir. Birinci boyut, projenin kendi hedeflerini; ikinci boyut ise portföyün kendi hedeflerini karşılayıp karşılamamasıdır. Bu bakımdan birinci boyut Geleneksel Proje Takibi ve Kontrol Süreçleriyle kolaylıkla gerçekleştirilebiliyorken, ikincisi için Kazanılmış Değer Analizi (Earned Value Analysis-EVA) ve Ürün Geliştirme Süreci (Stage-Gate Process) gibi çok iyi bilenen teknikten yararlanmayı gerektirmektedir (Levine, 2005: 24-25). Kısaca özetlemek gerekirse proje portföy yönetimi, işletmenin değerini maksimize eden, portföy dengesini gözeten ve nihai olarak işletmeninin genel iş stratejileriyle uyumlu olan projelerin seçilmesi sürecidir (Moustafaev, 2017: 8).

Birden çok projenin bir arada yönetilmesi ve koordinasyonu ise programdır. Programlar sayesinde çok karmaşık ve birbirine göre farklı konulara sahip çalışmalar yönetilebilmektedir.

PMI'ye göre, bir program "***bireysel olarak yönetilmesinden elde edilemeyecek faydalar elde etmek için koordineli bir şekilde yönetilen bir grup proje***"dir.

Program yönetimi, stratejik hedeflerin ölçülebilir iş sonuçlarına çevrilmesidir ve sonucun gerçekleşmesi için gereken birçok ilgili girişimin entegrasyonu ile birliktedir. Bir programın yönetiminden sorumlu olan kişiye genellikle program yöneticisi adı verilir. Program yöneticisinin rolü kuruma bağlı olarak değişir.

Program yönetimi, bir organizasyonun stratejik hedeflerini desteklemek ve büyük projeleri koordine etmek için kullanılan bir yönetim yaklaşımıdır. Programlar, organizasyonun bir dizi ilgili projeyi bir araya getirerek daha büyük bir hedefi veya sonucu elde etmesine yardımcı olur.

Program yönetimi aşağıdaki özelliklere sahip olmalıdır:

- Proje Koordinasyonu: Program yöneticisi, farklı projeleri koordine eder ve projeler arasındaki bağlantıları yönetir.

- Stratejik Hedeflere Katkı: Programlar, organizasyonun stratejik hedeflerine hizmet eder. Birden çok proje, organizasyonun daha büyük bir stratejik amaç için birleştirilir.

- Kaynak Paylaşımı: Programlar, projeler arasında kaynakların etkili bir şekilde paylaşılmasına yardımcı olur.

• Ölçülebilirlik: Program başarısı, organizasyonun genel stratejik hedeflerine nasıl katkı sağladığına dayalı olarak ölçülür. Programlar, organizasyonun stratejisini hayata geçirme yolunda önemli bir rol oynar.

• Risk Yönetimi: Program yönetimi, projelerin ve programın risklerini yönetir ve organizasyonun riskleri dengelemesine yardımcı olur.

Sonuç olarak, proje portföyü, organizasyonun birden çok projeyi izlediği ve yönettiği bir topluluk iken program yönetimi, organizasyonun stratejik hedeflerini desteklemek ve büyük projeleri koordine etmek için kullanılan bir yönetim yaklaşımıdır. Programlar, projelerin organizasyonun genel stratejik hedeflerine nasıl katkı sağlayabileceğini birleştirir ve izler.

Bazı işletmeler rolün ticari yönlerini vurgular. Bazı işletmeler ise, Bilişim Teknolojileri (BT) veya teknolojiye odaklanır; belirli teknik ve proje yönetimi niteliklerini vurgular. Program yöneticileri, yaklaşımın şekillendirilmesinden istenen sonuç kümesinin sunulmasına kadar çapraz işlevli programın uçtan uca ücretini yönlendirir. Program yöneticisi aşağıdakilerden sorumludur:

❖ Girişimlerin önceliklendirilmesi ve finanse edilmesi,

❖ Kuruluşlar arası bir yol haritası tanımlanması,

❖ Kaynak kapasitesinin ve kullanılabilirliğinin sağlanması,

❖ Projeler arasındaki bağımlılıkların yönetilmesi,

❖ Program düzeyinde hedeflere ulaşmasının sağlanması.

Program yöneticileri genellikle bir işletmenin Proje Yönetim Ofisi'ne (PMO) veya Stratejik Planlama Ofisine rapor verir ve bölümleri ve iş birimlerini kapsayan stratejik girişimleri yönetme sorumluluğunu taşır.

Portföy ve program yönetimi arasındaki farklar;

- Program yönetiminde birbirine benzer projeler, portföy yönetiminde ise benzer olmayan projeler veya farklı programlar yönetilmektedir.

- Program yönetiminin kapsamı, proje yönetiminden daha geniştir. Portföy ise işletmelerin stratejik hedefleri ile değişen organizasyon çapında bir kapsamı olmaktadır.

- Proje yöneticisi bir proje içinde birden fazla görevi yönetirken, program yöneticisi tüm projeler arasında koordinasyonu sağlar ve aynı amaç ve hedefe sahip olabilecek birbirleriyle ilişkili bağımlılıklara bakar. Genel olarak programlar daha uzun süre, bütçe, risk ve yüksek stratejik öneme sahiptir.

b. Proje Yönetim Organizasyonel Yapıları

Bilgi sistemleri denetçisi proje yönetim sürecine dahil olan grupların/bireylerin genel rollerine ve sorumluluklarına hâkim olmalıdır. Proje organizasyonuna bilgi sistemleri denetçisi de dahil edilebilir. Bilgi sistemleri denetçisi, sorumlu tarafların katılım seviyesini değerlendirebilmek için bağımsız bir inceleme gerçekleştirebilir. Bu durumda bilgi sistemleri denetçisi projede denetçi olarak değil danışman olarak görev alır. Organizasyon içerisindeki yetki ve kontrolü ana hatlarıyla belirten üç tür proje yönetimi organizasyon yapısı bulunmaktadır. Bu yapılara aşağıda kısaca değinilmektedir.

1. Fonksiyonel Yapılı Organizasyon

İş, fonksiyonel bazda bölümler arasında paylaştırılır. Proje yöneticisinin resmi olarak yönetim yetkisi bulunmayıp, sadece ekip üyelerine tanımlanacak faaliyetler konusunda tavsiyede bulunmaktadır.

Fonksiyonel yapılı organizasyonda, işler fonksiyonel bölümlere ayrılır. Bu yapının temel özelliği, organizasyondaki her bölümün, kendi alanında uzmanlaşmış olması ve fonksiyonel yöneticilerin, bu bölümlerin operasyonel yönetimini gerçekleştirmesidir. Proje yönetimi, daha çok fonksiyonel departmanların liderliğinde gerçekleşir ve proje yöneticisinin, proje süreci üzerinde sınırlı bir yetkisi bulunmaktadır.

Fonksiyonel yapılı organizasyonun başlıca özellikleri şunlardır:

- Proje yöneticisinin yetkisi sınırlıdır. Proje yöneticisi, ekip üyelerine sadece görevleri ve faaliyetleri yönlendirebilir, ancak kaynaklar üzerinde doğrudan kontrolü yoktur.

- İletişim ve koordinasyon genellikle fonksiyonel bölümler arasında yapılır, bu da bazen projelerde gecikmelere veya bilgi akışının zayıf olmasına neden olabilir.

- Proje sürecinde kaynakların planlanması ve tahsis edilmesi çoğunlukla fonksiyonel yöneticiler tarafından yapılır.

Fonksiyonel yapılı organizasyonun yapısı gereği avantajları ve dezavantajları bulunmaktadır. Fonksiyonel yapılı organizasyonun başlıca avantajları şunlardır:

- Fonksiyonel yapı, uzmanlaşmayı ve uzmanlık alanlarına derinlemesine odaklanmayı teşvik eder, bu da daha kaliteli çıktılar elde edilmesini sağlar.

- Çalışanlar kendi fonksiyonel yöneticileriyle daha fazla etkileşime girer, bu da uzmanlıklarının daha verimli bir şekilde kullanılmasına olanak tanır.

- Bu yapılar genellikle daha az karmaşık olabilir ve kaynakların yönetimi daha kolaydır.

Fonksiyonel yapılı organizasyonun başlıca dezavantajları şunlardır:

- Proje yöneticisinin sınırlı yetkisi, projelerdeki ilerleme hızını yavaşlatabilir. Çünkü kaynaklar ve bütçeler genellikle fonksiyonel yöneticiler tarafından kontrol edilir.

- Proje takımları, bölümler arası işbirliğini zorlaştırabilir, bu da projelerin tamamlanmasını engelleyebilir veya geciktirebilir.

- Fonksiyonel yapılar, proje yöneticilerinin karar alma yetkilerinin sınırlı olması nedeniyle esneklikten yoksun olabilir ve projelerin gereksinimlerine hızlı tepki verme yeteneğini kısıtlayabilir.

2. Proje Yapılı Organizasyon

İş proje bazında yürütülür. Proje yöneticisinin, proje bütçesi, proje programı, proje ekibi ve projede yer alan hususlara ilişkin resmi olarak yetkisi bulunmaktadır.

Proje yapılı organizasyon, projelerin yönetilmesinde daha fazla otonomi sağlayan bir yapıdır. Burada, projeler bağımsız olarak yönetilir ve proje yöneticisi, projeyle ilgili tüm kararları alma yetkisine sahiptir. Bu yapıda, proje yöneticisi sadece proje sürecinden değil, aynı zamanda proje bütçesi, kaynaklar, ekipler ve proje planlaması gibi tüm unsurlardan sorumludur.

Proje Yapılı organizasyonun başlıca özellikleri şunlardır:

- Proje yöneticisi, projeyle ilgili tüm kararları alır ve proje sürecinin her aşamasını yönetir.

- Proje ekibi, yalnızca projeye adanmış çalışanlardan oluşur ve proje yöneticisi tarafından yönetilir.

- Proje yöneticisi, proje sürecinde her türlü kaynağı, programı ve bütçeyi yönetir. Bu da projelere daha fazla kontrol sağlar.

Proje yapılı organizasyonun yapısı gereği avantajları ve dezavantajları bulunmaktadır. Proje yapılı organizasyonun başlıca avantajları şunlardır:

- Proje yöneticisi, proje sürecinin tamamında tam yetkiye sahip olduğu için daha hızlı kararlar alabilir ve projede esneklik sağlar.

- Bu yapılar, projeleri daha odaklı ve verimli hale getirir çünkü proje ekibi, yalnızca o projeye adanmış çalışandır.

- Kaynaklar doğrudan proje yöneticisi tarafından yönetildiği için, projelerin daha tutarlı ve etkili bir şekilde yürütülmesi sağlanabilir.

Proje yapılı organizasyonun başlıca dezavantajları şunlardır:

- Bu organizasyon yapısı, maliyetli olabilir çünkü her proje için ayrı bir proje ekibi oluşturulması gereklidir.

- Proje yöneticisinin tüm kaynaklar üzerinde mutlak bir kontrolü bulunması, diğer projelerle ilgili koordinasyonu zorlaştırabilir ve kaynakların verimli kullanılmasını engelleyebilir.

- Uzmanlık eksikliği olabilir çünkü proje bazlı yapılar, sürekli projelerle çalışan ekiplerin gelişmesini engelleyebilir.

3. Matris Yapılı Organizasyon

İş hem bölüm hem de proje bazında takip edilir. Proje yöneticisi ve bölüm yöneticileri arasında yönetim yetkisi paylaşılmaktadır.

Matris yapılı organizasyon, fonksiyonel ve proje bazlı yapıları birleştiren bir yapıdır. Bu organizasyonda, iş hem fonksiyonel bölümler hem de proje bazında takip edilir. Hem proje yöneticileri hem de fonksiyonel yöneticiler vardır ve her iki yönetici türü arasında yetki paylaşımı yapılır. Matris yapı, farklı yetkilerle daha esnek bir yönetim sağlar ve projelerin daha geniş bir perspektifle yönetilmesine olanak tanır.

Matris yapılı organizasyonun başlıca özellikleri şunlardır:

- Hem fonksiyonel yöneticiler hem de proje yöneticileri belirli bir çalışan üzerinde yetkiye sahiptir.

- Çalışanlar, hem fonksiyonel departmanlarına hem de proje ekibine hizmet eder.

- Yönetim daha karmaşık olabilir çünkü iki farklı yöneticinin beklentileri ve gereksinimleri arasında denge kurmak gereklidir.

Matris yapılı organizasyonun yapısı gereği avantajları ve dezavantajları bulunmaktadır. Matris yapılı organizasyonun başlıca avantajları şunlardır:

- Kaynaklar daha etkin kullanılabilir çünkü çalışanlar hem fonksiyonel hem de proje ekibine hizmet eder.

- Çalışanlar, proje ekibiyle işbirliği yaparken aynı zamanda fonksiyonel bilgi ve becerilerinden faydalanabilirler.

- Proje yöneticisinin yetkisi daha fazla olduğundan, projelerin daha verimli yönetilmesi sağlanabilir.

Matris yapılı organizasyonun başlıca dezavantajları şunlardır:

- Karmaşık bir yapı olabilir ve iki yöneticinin olduğu durumlarda çatışmalar veya ikili yönetim problemleri yaşanabilir.

- Çalışanlar, iki yöneticinin taleplerini dengelemek zorunda kalabilir, bu da stres yaratabilir ve iş verimliliğini etkileyebilir.

- Koordinasyon ve iletişim sorunları yaşanabilir, çünkü proje ve fonksiyonel yöneticilerin arasında sürekli bir iletişim ve denetim gereklidir.

Sonuç olarak, fonksiyonel, proje bazlı ve matris yapılı organizasyonlar, organizasyonel yapıların proje yönetimine yaklaşımını farklı şekillerde yansıtan önemli modellerdir. Her bir yapının avantajları ve dezavantajları, organizasyonun ihtiyaçlarına göre dikkatlice değerlendirilmelidir. Projelerin daha etkin yönetilebilmesi için doğru organizasyon yapısının seçilmesi, başarılı bir proje yönetimi için kritik öneme sahiptir.

c. Proje Paydaşlarının Rol ve Sorumlulukları

Projede yer alan grup/bireylerin rol ve sorumlulukları aşağıda belirtilmektedir.

Proje Yönlendirme Komitesi

Proje yönlendirme komitesi, projeye dair tüm stratejik kararların alındığı ve yönlendirildiği bir yapıdır. Bu komite, proje kapsamında belirlenen hedeflere ulaşılabilmesi için genel yönlendirmeyi sağlar, yani projenin başarıyla tamamlanabilmesi için gereken kararları alır, sorunları tespit eder ve çözüm önerileri sunar. Proje sponsoru komiteye başkanlık eder ve komite, proje takibinden, maliyet

kontrolünden, zaman çizelgesinin denetlenmesinden ve proje risklerinin yönetilmesinden sorumludur. Komite ayrıca, proje ekibinin hedeflere ulaşabilmesi için gerektiğinde koordinasyon sağlar ve düzeltici faaliyetler önerir. Bu süreç, proje yönetiminin her aşamasında stratejik kararların alınmasını, proje hedefleri doğrultusunda gerekli düzeltici eylemlerin uygulanmasını ve projenin zamanında tamamlanmasını sağlar. Günümüz projelerinde, komite dijital araçlar ve analitik platformlar kullanarak projelerin izlenmesini ve değerlendirilmesini daha etkili hale getirir. Bu araçlar, gerçek zamanlı veri sağlayarak komitenin daha hızlı ve doğru kararlar almasına olanak tanır. Bilgi sistemleri denetçisi, komitenin projeye katkısını değerlendirirken, dijital çözümlerin entegrasyonu ve veri odaklı karar alma süreçlerini gözlemlemelidir. Ayrıca, komite üyelerinin düzenli olarak güncellenen veri setlerini kullanarak stratejik kararlar alması, projede ortaya çıkabilecek riskleri proaktif bir şekilde tespit etmeye ve bu risklere karşı önlem almaya olanak tanır.

Üst Düzey Yönetim

İşletme içerisinde mevcut kaynakların kullanılması için daima bir rekabet ortamı vardır. Proje sürecinde var olan yüksek belirsizlik oranı projeyi bu yarışın dışında bırakabilir. Projeye üst düzey yönetiminin vereceği destek, projenin sürekliliği ve başarısı ile proje ekibi tarafından proje amaç ve öneminin anlaşılmasında önemli rol oynamaktadır. Bu farkındalık ve üst yönetim desteği krizler ve çatışmaları önlemekte ve belirsizlikleri ortadan kaldırmaktadır.

Üst düzey yönetim, bir organizasyonun proje hedeflerine ulaşması için kaynak ve liderlik sağlar. Günümüzde, üst yönetim kararlarını dijital paneller ve iş zekası araçlarıyla alarak, projelerin her aşamasını daha şeffaf bir şekilde izleyebilmektedir. Bu araçlar, proje ilerlemesini, bütçe takibini ve zaman çizelgesindeki sapmaları analiz eder, böylece yönetimin doğru zamanda müdahale etmesine olanak tanır. Bilgi sistemleri denetçisi, üst yönetimin proje stratejileriyle uyumlu olup olmadığını denetlerken, dijital araçların karar alma süreçlerine nasıl entegre edildiğini değerlendirmelidir. Üst yönetim, aynı zamanda yapay zeka ve makine öğrenimi teknolojilerini kullanarak proje yönetimi süreçlerini optimize edebilir. Bu tür araçlar, üst düzey yönetimin projelerin gelecekteki yönü hakkında daha hızlı ve etkili kararlar almasına yardımcı olur.

Proje Sponsoru

Proje sponsoru, projeyi finansal ve kritik noktalarda temsil eden, projeyi stratejik olarak yönlendiren üst düzey yöneticidir. Proje sponsoru, işletmenin üst yönetimi veya genel müdüründen projeye finansal ve idari desteği alır. Bu nedenle, proje sponsoru, proje yöneticisi ve proje lideri (varsa) ile sürekli iletişim halinde olmalı, proje ilerlemesi hakkında düzenli bilgi almalı ve proje kararlarıyla ilgili rehberlik sağlamalıdır. Proje sponsoru, proje yönetimi ekibiyle işbirliği yaparak, proje finansmanı, faaliyet alanının belirginleşmesi ve üçüncü kişilerin projeden faydalanması gibi konularda stratejik yönlendirme sağlar.

Projelerin başarısı, güçlü bir liderlik ve stratejik yönlendirme gerektirir. Bu liderlik, proje sponsorunun sağladığı finansal ve idari desteği içerir. Günümüzde, proje sponsorları dijital araçları ve teknoloji tabanlı çözümleri daha etkin bir şekilde kullanarak proje yönetim süreçlerine entegre etmektedir. Bu araçlar, proje performansının izlenmesi, finansal desteklerin sağlanması ve kaynakların gerçek zamanlı olarak yönetilmesi konusunda önemli kolaylıklar sunar. Proje sponsorları, bulut tabanlı proje yönetim yazılımları sayesinde bütçe takibini etkin hale getirir ve projenin finansal sağlığını sürekli izler. Bilgi sistemleri denetçisi, proje sponsoru tarafından sağlanan finansal ve idari desteğin etkinliğini değerlendirirken, dijital yönetim araçlarının karar süreçlerine nasıl katkı sağladığını da incelemelidir. Ayrıca, gelişmiş analitik araçlar kullanılarak, risk yönetimi ve yatırım getirisi (ROI) hesaplamaları daha doğru ve öngörülebilir sonuçlar elde edilmesini sağlar.

Kullanıcı Yönetimi

Kullanıcı yönetimi, proje ekibinin faaliyetlerini (gereksinimlerin tanımlanması, tasarım, test senaryolarının tanımlanması, kabul testi, kullanıcı eğitimi vb.) takibinden sorumludur.

Modern projelerde, kullanıcı yönetimi süreçleri dijital platformlar ve anket araçları kullanılarak daha verimli hale getirilmektedir. Bu platformlar, kullanıcı geri bildirimlerini anında toplar ve proje ekibine raporlar, böylece proje süreçleri sırasında kullanıcı ihtiyaçlarına hızlı bir şekilde yanıt verilir. Bilgi sistemleri denetçisi, kullanıcı gereksinimlerinin projeye entegrasyonunu izlerken, dijital araçların

kullanıcı geri bildirimlerini toplama ve değerlendirme süreçlerine nasıl katkı sağladığını gözlemelidir. Kullanıcı yönetimi, ayrıca proje sürecinde yapay zeka ve kullanıcı deneyimi (UX) test araçları kullanarak, daha kullanıcı dostu sistemlerin geliştirilmesini sağlayabilir.

Kullanıcı Proje Ekibi

Kullanıcı proje ekibi, projede çalışan, projenin başarısına katkıda bulunan kişilerdir. Tüm ekip üyeleri almış oldukları görevlerin durumlarının proje yöneticisine raporlanmasından sorumludur.

Günümüzde, kullanıcı proje ekibi üyeleri genellikle çevik (agile) metodolojilere dayalı proje yönetim tekniklerini kullanarak, daha esnek ve hızlı bir şekilde projeye katkı sağlarlar. Bilgi sistemleri denetçisi, kullanıcı proje ekibinin etkinliğini izlerken, çevik metodolojilerin ve modern proje yönetim araçlarının (örneğin Jira, Trello gibi araçlar) kullanımını gözlemlemelidir. Bu araçlar, ekip üyelerinin görevlerini kolayca takip etmelerini ve düzenlemelerini sağlar. Ayrıca, bu ekiplerin rol ve sorumluluklarının net bir şekilde tanımlanması, projenin hızlı bir şekilde ilerlemesini sağlar. Kullanıcı proje ekibinin başarılı olabilmesi için, ekip üyelerinin açık iletişim kurarak, proje hedeflerine odaklanmaları gerekir.

Proje Yöneticisi

Proje yöneticisi, projenin amaç ve hedeflerine ulaşmasının sağlanmasından sorumludur. Bir proje yöneticisinin başarısı, gereksinim ve beklentilerin doğru tanımlanması, ulaşılabilir hedeflerin belirlenmesi, etkili ve açık iletişim kurulması, kapsam, maliyet, zaman ve kalite kısıtları dahilinde planlamanın etkin ve verimli şekilde yapılabilmesine bağlıdır.

Başarılı bir proje, yalnızca doğru yönetimle değil, aynı zamanda hedeflere ulaşmak için stratejik bir yönlendirme ile gerçekleşir. Günümüz projelerinde, proje yöneticileri, modern yazılım ve proje yönetim araçları kullanarak projeyi etkin bir şekilde yönetir. Proje yöneticisinin temel görevleri, projeyi zamanında ve bütçe dahilinde tamamlamak, ekip üyeleriyle açık iletişim sağlamak ve riskleri yönetmektir. Bilgi sistemleri denetçisi, proje yöneticisinin bu görevleri yerine getirirken dijital araçlardan ne derece faydalandığını izlemelidir. Yapay zeka destekli tahminleme araçları ve proje yönetimi yazılımları, proje yöneticisinin daha bilinçli kararlar almasına yardımcı olabilir. Bu araçlar, zaman çizelgesi yönetimini, bütçe izlemeyi ve kaynak tahsisini optimize etmeye olanak tanır.

Proje yöneticisinin belirlenmesi sırasında aşağıda sıralanan nitelikler aranır:

- Altyapı ve Deneyim: Proje yöneticisinin proje yönetimi konusunda yeterli altyapı ve deneyime sahip olması gerekmektedir.

- Liderlik ve Stratejik Uzmanlık: Proje yöneticisi, proje ekibi üyelerinin doğrudan saha yöneticisi olmayabilir. Farklı bölümlerden proje ekibine katılan kişilerin yönetilebilmesi için etkin liderlik göstermesi beklenir. Proje yöneticisi, projenin kurumun stratejik planları ile bağlantılı gitmesine de özen göstermelidir.

- Teknik Uzmanlık: Proje yöneticisinin proje konusunda teknik uzman olması gerekmemektedir. Ancak proje sırasında olup biteni takip edebilecek, doğru soruları soracak ve verilen cevapları anlayacak derecede konuya hakim olmalıdır.

- İletişim Becerisi: Proje esnasında proje yöneticisi kendi ekibiyle, diğer ekiplerle ve üst yöneticilerle yoğun iletişim içinde olacaktır. Bu iletişimdeki etkinlik projenin başarısı için çok önemlidir.

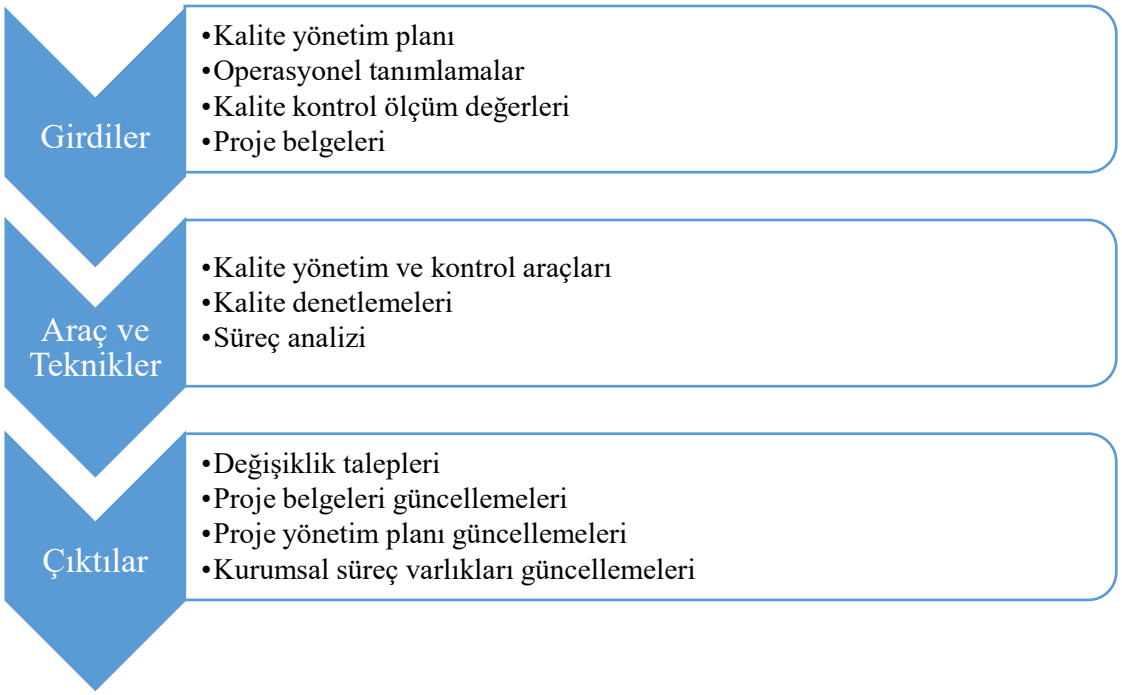
- Yönetsel Beceri: Proje yöneticisinin stratejik planlama, bütçe, insan kaynakları yönetimi, kalite yönetimi vb. birçok konuda yeterli bilgi ve birikimi bulunmalıdır.

Kalite Güvence

Kalite güvence, proje çıktılarının incelenerek gereksinimlere uygunluğunun takibinin yapılması ve ilgili organizasyon süreçlerinin teslimatından sorumludur. Proje esaslarına bağlı kalarak işletmenin kalite yönetim sistemini uygular; kalite yönetim planı, kalite güvencesinin sağlanması ve kalite kontrolü gibi alt süreçler içerir. Kalite yönetim planı ve kontrol sonuçları kalite planlama araç ve teknikleri kullanılarak kalite sistemi uygulanır. Kalite planlama, proje ile ilgili olan kalite standartlarının

belirlenmesi ve bu standartların nasıl sağlanacağının tanımlanması sürecidir. Kalite kontrolü ise proje sonuçlarının, kalite standartlarına uygunluğunun kontrol edildiği süreçtir. Aynı zamanda, kalite arttırıcı faaliyetler de kalite kontrol sürecinin bir parçasıdır.

Bir kalite güvence uygulamasının yapılması aşağıdaki gibidir:



Kalite yönetimi ve proje yönetimi arasında benzer özellikler bulunmaktadır. Aşağıda sıralanan kavram ve yaklaşımlar her ikisi için de kullanılabilir:

- Müşteri Memnuniyeti: Proje, müşterinin ihtiyaçlarının giderilmesi konusundaki beklentilerini söz verilen biçimde karşılamalıdır.

- Önlem Alma: Kalite proje içinde planlanır ancak denetlenmez. Hatanın önlenmesi düzeltilmesinden her zaman daha az maliyetlidir.

- Sorumlulukları Yönetme: Başarıya ulaşmanın proje ekibi üyelerinin tümünün katılımı ve kaynakların optimum şekilde kullanımı ile münkün olduğunu vurgular. Ekip üyelerinin rol ve sorumlulukları tanımlanmalı, gerekli kaynaklar sağlanmalıdır.

- Süreçleri Safhalara Ayırma: Her bir sürecin planla, yap, kontrol et ve gerçekleştir safhalarından oluşması gerektiğini vurgular. Bu döngü başarıya ulaşılıncaya kadar tekrarlanmalıdır.

- Bazı işletmelerde kalite faaliyetleri kalite güvence bölümü tarafından yapılmaktadır. Kalite güvence, hataların bulunması ve giderilmesi ile ilgilidir.

İki tür kalite güvence vardır:

- İç Kalite Güvence: Yönetime ve proje ekibine sağlanan güvencedir. İç kalite güvence, bir işletmenin veya proje ekibinin iç süreçlerine ve uygulamalarına odaklanır. Bu, şirket içinde kaliteyi sağlamaya yönelik bir dizi önlemi içerir. İşte iç kalite güvence ile ilgili bazı özellikler:

❖ Kalite standartlarının ve prosedürlerinin geliştirilmesi ve uygulanması.

❖ Sürekli iyileştirme süreçlerinin yürütülmesi ve izlenmesi.

❖ Kalite denetimleri ve iç denetimlerin düzenlenmesi.

❖ Personelin kalite standartlarına uyması için eğitilmesi ve bilinçlendirilmesi.

❖ İşletme içinde hataları tespit edip giderme süreçlerinin oluşturulması.

- Dış Kalite Güvence: Proje müşterilerine sağlanan güvencedir. Müşterilere veya dış paydaşlara kaliteyi garanti etmeye odaklanır. Ürün veya hizmetin müşteri gereksinimlerine ve kalite standartlarına uygun olduğunun doğrulanması, müşteri geri bildirimlerinin toplanması ve değerlendirilmesi, ürün veya hizmetin dış denetimler ve sertifikasyonlar aracılığıyla bağımsız olarak doğrulanması, müşteri memnuniyetini artırmak için sürekli geri bildirim ve iyileştirme süreçlerinin uygulanması gibi önemli adımları içermektedir.

İşte dış kalite güvence ile ilgili bazı özellikler:

Kalite güvence uygulamasının yapılması sürecinin, proje yöneticisi ve proje ekibi tarafından hazırlanması gereken üç temel girdisi bulunmaktadır:

- Kalite Yönetim Planı: Proje ekibinin kalite politikasını nasıl uygulayacağı tanımlayan plandır.

- Kalite Kontrol Ölçüm Değerleri: Bu değerler kalite kontrol testlerinden sağlanmaktadır. Değerler karşılaştırılabilir ve analiz edilebilir nitelikte olmalıdır.

- Operasyonel Tanımlamalar: Projenin süreçlerini tanımlayan ölçütler, bunların değerleri ve ölçüm birimleri kalite güvence için gerekmektedir. Kalite güvence gerekliliklerini kalite güvence bölümü veya bazı durumlarda proje yöneticisi hazırlamaktadır. Kalite güvence için maliyet kazanç analizi, akış diyagramı, deney tasarımı vb. gibi yaklaşımlar kullanılmaktadır.

Kalite kontrolleri öğrenme amaçlıdır. Kalite kontrollerindeki temel fikir proje süresince oluşan hatalardan öğrenilenlerin, bu proje ve ilerleyen diğer projeler için işletme içinde içselleştirilerek hayata geçirilmesiyle hataların tekrarını önlenmektedir. Kalite kontrolleri proje süresince önceden belirlenmiş sıklıklarla veya haber verilmeden de yapılabilmektedir. Kalite kontrollerini, iç denetçiler yapabileceği gibi dış kaynaklardan da hizmet alımı yapılabilmektedir.

Proje ekibinin kalite kontrol için kullanacağı bazı beceriler ve teknikler aşağıdaki şekilde sıralanmaktadır:

- İstatistiksel kalite kontrol teknikleri,

- Hatalı ürünlerin müşteriye ulaşmasına engel olacak kontroller,

- Kalitede anormalliklere ve dalgalanmalara yol açan nedenlerin tespiti,

- Kalitenin istenen seviyede olması için gerekli kontrol sınırlarının belirlenmesi.

Sistem Geliştirme Yönetimi

Sistem geliştirme yönetimi, proje kapsamında gereksinim duyulan sistem/uygulamanın geliştirilerek, işlerliğinin sağlanması, stratejik BT bakış açısıyla uyumluluğun güvencesinin verilmesi, bakım hizmetleri gibi teknik destek faaliyetlerinin yürütülmesinden sorumludur.

Sistem geliştirme yönetimi, yalnızca yazılımın işlevsel gereksinimlere uygun şekilde geliştirilmesini sağlamaz, aynı zamanda bu sistemin stratejik BT hedefleriyle uyumlu olmasını da denetler. Teknolojik stratejiler, iş ihtiyaçlarını karşılamak için belirlenir ve bu stratejilerin projeye entegrasyonu dikkatle izlenir. Ayrıca, sistemin sürekli iyileştirilmesi için bakım ve destek hizmetleri sağlanarak, projenin sürdürülebilirliği ve verimliliği artırılır. Bu süreç, sadece yazılım geliştirmeyi değil, aynı zamanda işletme hedefleriyle uyumlu, gelişen BT stratejilerinin de projeye entegrasyonunu içerir.

Sistem Geliştirme Proje Ekibi

Sistem geliştirme proje ekibi, projede sistem geliştirme yönetimi tarafından atanan görevlerin yürütülmesinden sorumludur.

Proje sürecinde yazılım geliştirmeyi yönlendiren ekip, her aşamada belirli görevleri yerine getirmek için uzmanlık alanlarında yoğunlaşır. Bu ekip, yazılımın tasarımından kodlamaya, testlerden entegrasyona kadar her adımda aktif rol alır. Kullanıcı gereksinimlerine odaklanarak, bu gereksinimlere uygun çözümler geliştirmek ekip üyelerinin sorumluluğundadır. Ayrıca, yazılım geliştirme sürecinde karşılaşılan hataları çözmek, yazılım performansını artırmak ve son kullanıcı için en iyi çözümü sağlamak amacıyla sürekli iyileştirmeler yapılır.

Güvenlik Yöneticisi/Güvenlik Ekibi

Güvenlik yöneticisi veya güvenlik ekibi, sistem kontrollerinin ve destek süreçlerinin, kurumsal güvenlik bakış açısına uygun şekilde tesis edilmesinden sorumludur.

Organizasyonun bilgi güvenliği stratejilerinin belirlenmesi ve uygulanması, güvenlik yöneticisi ve güvenlik ekibinin en önemli sorumlulukları arasındadır. Bu ekip, sistemin tüm bileşenlerinin güvenliğini sağlamak amacıyla çeşitli önlemler alır ve uygulamalarını sürekli olarak denetler. Güvenlik yöneticisi, aynı zamanda güvenlik politikalarının oluşturulması ve organizasyona entegrasyonu gibi kritik süreçleri yönetir. Güvenlik ekibi, tehditleri tespit etme, bu tehditlere karşı önleyici tedbirler alma ve olası güvenlik ihlallerine hızla müdahale etme konusunda uzmandır. Ayrıca, ekip, sistemdeki her bileşenin güvenliğine dair standartlar belirler ve bu standartlara uyumu denetler.

Bilgi Sistemi Güvenlik Mühendisi

Bilgi sistemi güvenlik mühendisi, güvenlik açıklarının belirlenmesi ve bu açıklıklara ilişkin riskin en aza indirilmesi veya sınırlamak için bilimsel ve mühendislik ilkelerini uygulamaktadır. Bu rolü yerine getirmenin anahtarı hem geniş alanda savunulması hem de derinlemesine güvenlik ilkelerine göre ağ, ortam ve uygulama yapılarını inşa edilmesi için ihtiyaçların, gereksinimlerin, mimarilerin ve tasarımların tanımlanmasıdır.

Bir organizasyonun dijital güvenliğini sağlamak, sürekli değişen tehditlere karşı korunma sağlamak ve güvenlik stratejilerini geliştirmek, bilgi sistemi güvenlik mühendisinin en önemli görevlerindendir. Güvenlik mühendisi, organizasyonun tüm bilgi sistemlerinin güvenliğini sağlamak için kapsamlı bir yaklaşım benimser, riskleri minimize etmek ve sistemleri güvenli tutmak adına çeşitli teknik ve mühendislik yöntemlerini uygular. Bu roldeki profesyonel, organizasyonun dijital varlıklarını koruyarak, güvenlik altyapısının her aşamasında kritik kararlar alır ve güvenlik stratejilerini sürekli olarak günceller.

d. Proje Yönetim Ofisi (PMO)

Proje Yönetim Ofisi, projeyle ilgili yönetişim süreçlerini standartlaştıran, kaynak, metodoloji, araç ve tekniklerin paylaşılmasını kolaylaştıran bir yönetim yapısıdır. PMO, yürürlükteki standart ve prosedürlerin korunması, geliştirilmesine yönelik yeterli kaynağa sahip olmalıdır. PMO'nun amacı yönetilen süreçlerin kalitesinin iyileştirilmesi ve başarının güvenceye alınmasıdır. PMO proje/program içeriğinden daha çok faaliyet/görevlere odaklanır.

Bilgi sistemleri denetçisi, denetime ilişkin projelerin içeriğinin yanı sıra prosedüre ilişkin yönleri de değerlendirmelidir.

Proje portföy yönetiminin hedefleri aşağıdakilerden oluşmaktadır:

- Bireysel olmayan proje portföy sonuçlarının iyileştirilmesi,
- Projelerde önceliklendirme ve zaman planlamasının yapılması,
- İç ve dış proje kaynak koordinasyonlarının sağlanması,
- Proje süresince bilgi akışının sağlanması.

Proje portföy yönetimi için bir proje portföy veritabanı zorunludur. Veritabanı, projelerin sahiplik, program, hedef, tür, durum ve maliyet gibi verileri içermelidir. Çubuk grafik, kar ve risk matrisi ve proje portföyü gibi ilerleme grafikleri gibi tipik proje portföy raporları proje portföy yönetimi kapsamında hazırlanmaktadır.

e. Proje Faydalarının Tanımlanması

Proje faydalarının gerçekleştirilmesinin amacı, BT'nin ve iş birimlerinin değer yönetimi sorumluluklarının yerine getirilmesini sağlamaktır. Hem programa hem zamana duyarlı pazara, sektör ve yasal gerekliliklere göre zamanında, bütçe dahilinde bulunan yetenekler daha büyük katkı sağlamaktadır. Söz konusu BT hizmetleri ve varlıkları iş değerine de katkıda bulunmaktadır.

Projelerin sağladığı faydaların belirlenmesi, başarıyla tamamlanan her projenin ardından elde edilen değerlerin hem nicel hem de nitel açıdan analiz edilmesi sürecidir. Bu süreç, BT altyapısının iş

birimleriyle uyumunu ve organizasyonun stratejik hedeflerine ulaşmasını sağlamayı hedefler. BT hizmetleri ve varlıkları, iş süreçlerine katkı sağlarken, operasyonel maliyetlerin azaltılmasına ve hizmet kalitesinin iyileştirilmesine de yardımcı olur. Proje faydaları yalnızca teknik başarılarla sınırlı kalmaz, aynı zamanda iş süreçlerindeki gelişmeler, müşteri memnuniyeti artışı ve gelir artışı gibi iş değerlerine de katkı sağlar. Bu nedenle, proje faydalarının doğru bir şekilde tanımlanması, BT departmanlarının verimliliği ve genel organizasyonel hedeflerin gerçekleşmesini doğrudan etkiler.

1.1.2. Proje Planlanma Metotları

Proje planlaması kapsamında, proje yöneticisi tarafından proje kapsamı (paydaşlar ile mutabakat sağlanarak), hedeflenen iş uygulamasının gerçekleştirilebilmesi için yapılacak görevlendirmeler, görevlerin sırası, düzeni, süresi ve önceliği, BT ve BT dışı kaynaklar, bütçe ve maliyet, proje kapsamındaki (işçilik, hizmet, materyal, tesis ve ekipman vb.) giderlerin fonlama kaynağı belirlenmelidir.

Doğru kararların alınması ve buna uygun faaliyetlerin yürütülebilmesi için projelerin mutlaka planlama aşamasından geçmesi gerekmektedir. Planlama, proje süresi boyunca yalnızca bir kere yapılan ve proje sonuna kadar hiç değişikliğe uğramadan uygulanan bir nitelik taşımaz. Proje planı, proje süresince yapılan kontroller sonucunda elde edilen bilgiler ve değişen dış etkenler uyarınca, devamlı olarak değerlendirme, gözden geçirme ve yenilenme süreçleri içerisinde bulunmaktadır. Böylece plan, zamanla uygulanamaz bir nitelik kazanmak yerine, sürekli güncellenerek proje tamamlanana kadar uygulanabilirliğini korumaktadır. Planlama yapılmadığı takdirde gelecekteki fırsatları ve tehlikeleri görmek mümkün olmayacağından, bu konuda gerekli önlemler de alınamayacaktır (Barutçugil, 1984:162). Neyin, niçin, nasıl ve ne zaman yapılacağını tanımlayan, projedeki işlerin yürütülmesini ve projedeki çalışanların yönetimini sağlayan planlama çalışmaları yapılmaksızın, projenin başarılı bir şekilde yürütülmesi ve sonuçlandırılması mümkün değildir. Proje planının geliştirilmesinde, görev ve sorumlulukların belirlenmesi, proje zaman cetvelinin hazırlanması ve proje bütçesinin çıkarılması en önemli çalışmalar arasındadır (Barutçugil, 1988:239-240).

Proje planlamada nihai hedef üç farklı şekilde olabilir;

- Eldeki kaynaklar çerçevesinde projenin en kısa zamanda bitirilmesi,

- Daha önceden belirlenmiş bir proje süresi dahilinde en az kaynak kullanımı ile projenin bitirilmesi,

- Proje toplam maliyetini en az yapacak bir proje süresinde projenin bitirilmesi.

Projenin planlama aşaması, belirlenen amaçlara göre projeyi teşkil eden faaliyetlerin birbirleriyle olan mantıksal ilişkileri, tamamlanma süreleri, kaynak gereksinimleri ve maliyetleri göz önünde bulundurularak gerçekleştirilir. Dolayısıyla projenin planlama aşamasında gereksinim duyulan bilgi ve verilerin elde edilme sürecinde çalışmaya ışık tutacak soruların sorulması önemlidir. Örneğin; "*Proje ne gibi faaliyetlerden oluşmaktadır?*", "*Faaliyetlerin birbirleriyle olan ilişkileri nasıldır?*", "*Her faaliyetin ne tür bilgi sistemi gücüne gereksinimi vardır?*", "*Her faaliyetin hangi miktarda işgücüne ihtiyacı vardır?*", "*Faaliyetlerin ve projenin bütününün maliyet unsurları nelerdir?*" gibi soruların sorulması ve cevap bulması bir projenin etkin bir şekilde planlanabilmesi için gerekmektedir.

Özet olarak, planlama safhasında proje, ana faaliyet gruplarına ayrılır ve her grup içindeki temel faaliyetler tespit edilir. Her bir faaliyetin tamamlanma süresi ve gerektirdiği kaynak miktarı belirlenir. Daha sonra faaliyetlerin birbirleriyle olan mantıksal ilişkileri göz önünde bulundurularak faaliyet sıraları belirlenerek tüm proje bu mantıksal ilişkiler ve faaliyet öncelik sıralaması ışığında bir bütün olarak ifade edilir. Böylece proje şebekesi kurularak proje programlama aşamasına zemin hazırlanmış olur.

Proje programlama, kaynak gereksiniminin ve tahmin edilen süre içinde projenin gidişatının programlanmasıdır. Proje programlamada ilk aşama, her bir faaliyet için gerekli süreyi belirlemektir (Grow, 1975:185). Ayrıca bu aşamada, her faaliyetin başlama ve bitiş zamanını gösteren bir zaman diyagramı hazırlanmaktadır. Proje programı, proje açısından önem arz eden kritik faaliyetleri göstererek, faaliyetlerin serbestlik süresi ve gecikme miktarı hakkında bir fikir vermelidir (Monks, 1996:352; Halaç, 1995:184). Bu şekilde projenin zamanında bitirilebilmesi için dikkatle takibi gereken ve zaman açısından kritik olan faaliyetlerin tanımlamaları yapılmaktadır. Ayrıca kritik olmayan

faaliyetlerin sahip oldukları boş zamanların tespiti sınırlı kaynakların mümkün olduğu kadar projenin daha kritik olan işlemlerine dağıtılabilmesini sağlamaktadır.

Programlama, her faaliyetin yalnızca ne zaman tamamlanması gerektiğini göstermez, aynı zamanda "*boşluk zamanları*" da göz önüne alınarak, bir faaliyetin "*en erken başlama*", "*en geç başlama*", "*en erken bitirme*", "*en geç bitirme*" tarihlerini de belirtmektedir. Böylece var olan insan gücü ve kaynaklardan birlikte yararlanacak faaliyetler arasında sıralamalar yapmak suretiyle, sınırlı insan gücü ve kaynaklardan en çok faydalanma gerçekleştirilmiş olur.

Bir proje programı aşağıdaki unsurlar hakkında bilgi vermelidir (Jensen, 1994):

- Proje aşamalarının ayrıntıları,

- Faaliyet sayısı,

- Faaliyet süreleriyle ilgili minimum ve maksimum tahminler,

- Hedef süreyle ilgili kısıtlamalar ve mantıksal ilişkiler,

- Kaynak ve maliyet kısıtlamaları,

- İyileştirme yöntemleri.

Programlama aşamasında, projenin hedeflenen süre içerisinde bitirebilmenin mümkün olup olmadığı belirlenebilmekte ve proje programının ön gördüğü tamamlanma süresinin kısaltılması ihtiyacı doğduğu zaman hangi faaliyetlere hızlandırma işlemi uygulanması gerektiği ve bu işlemi gerçekleştirmenin maliyeti tespit edilebilmektedir.

Proje programının sağladığı avantajlar aşağıdaki gibi ifade edilebilir (Monks, 1996):

- Tüm projeyi ve birbirleriyle ilişkili faaliyetleri koordine etmektedir.

- Tüm faaliyetlerin mantıklı bir biçimde planlanmasını sağlayarak faaliyetlerin organize edilmesini kolaylaştırmaktadır.

- Öncelik ilişkilerini ve özelikle kritik olan faaliyetlerin sırasını tanımlamaktadır.

- Gerçek değerlerle karşılaştırma yapabilmek için projenin tamamlanma süresinin (veya maliyetinin) tahminiyle ve bu konudaki standartlarla ilgili bilgileri sağlamaktadır.

- İnsan gücü, malzeme ve mali alanda yer değiştirebilecek kaynakları tanımlayarak kaynakların daha iyi kullanılmasını sağlamaktadır.

Projenin programlama aşaması tamamlandıktan sonra projenin kritik yolu üzerinde önemle durulması gerekmektedir. Proje yöneticisinin projeyi olası en kısa sürede ve en düşük maliyetle tamamlayabilmesi için kritik yolun üzerindeki faaliyetlere hızlandırma işlemi uygulanması ve projenin tamamlanma süresinin hedef süreye uygun şekilde ayarlanması gerekmektedir.

1.1.2.1. Proje Kapsamının Belirlenmesi ve Fizibilite Çalışması

Proje başlangıcındaki onay sonrasında ihtiyacı ve bu kapsamdaki alternatif çözümleri belirlemek amacıyla fizibilite analizi gerçekleştirilmektedir. İş olurluk incelemesi genelde fizibilite analizine dayanmaktadır. Üretilen çözümün ihtiyaca ve tanımlanan bütçe, zaman kısıtlarına uygunluğunu analiz edebilmek amacıyla gerçekleştirilen ön çalışmadır.

Fizibilite çalışmaları, bir projenin veya işletmenin başarı potansiyelini ve uygulanabilirliğini değerlendirmek için önemli bir adımdır. Aşağıda fizibilite çalışmalarının faydalarını maddeler halinde ele alınmıştır:

Projeyi Değerlendirme: Fizibilite çalışmaları, bir proje veya işletme fikrinin gerçekleştirilebilirliğini değerlendirerek, projenin başarılı olma potansiyelini belirlemeye yardımcı olur.

Riskleri Tanımlama: Fizibilite çalışmaları, projenin karşılaşabileceği potansiyel riskleri ve zorlukları tespit etme olanağı sunar. Bu, risk yönetimi stratejilerinin geliştirilmesine yardımcı olur.

Finansal Durumu İnceleme: Fizibilite çalışmaları, projenin maliyetlerini ve potansiyel getirilerini değerlendirerek, finansal açıdan projenin sürdürülebilirliğini ve karlılığını değerlendirmeye yardımcı olur.

Kaynakların Yeterliliği: Bir projenin başarılı olabilmesi için gerekli kaynakların (finansal, insan kaynakları, teknoloji vb.) mevcut olup olmadığını belirleme imkanı sunar.

Pazar Araştırması: Fizibilite çalışmaları, hedef pazarı ve müşteri ihtiyaçlarını daha iyi anlamaya yardımcı olur. Pazarın büyüklüğü, rekabet ortamı ve potansiyel müşterilerin tepkileri hakkında bilgi sağlar.

Karar Verme Sürecine Destek: Fizibilite çalışmaları, projenin veya işletmenin gerçekçi bir şekilde değerlendirilmesini ve yöneticilere bilinçli kararlar verme olanağı sunar.

Yatırımcıların ve Paydaşların Güvenini Kazanma: Fizibilite çalışmaları, yatırımcıların ve diğer paydaşların projeye olan güvenini kazanmada önemli bir rol oynar. İyi bir fizibilite raporu, finansman sağlama sürecini kolaylaştırabilir.

Zaman ve Kaynakların Etkin Kullanımı: Fizibilite çalışmaları, potansiyel olarak başarısız olabilecek projelerin erken aşamada tanımlanmasına yardımcı olur, böylece zaman ve kaynak kayıpları önlenir.

Alternatif Çözüm Yollarını İnceleme: Fizibilite çalışmaları, farklı proje yaklaşımlarını ve alternatif çözüm yollarını değerlendirme fırsatı sunar, böylece en uygun stratejinin seçilmesine yardımcı olur.

Proje Yönetimi İçin Temel Bilgiler: Fizibilite çalışmaları, proje yönetimi süreçlerini daha iyi anlamaya ve planlamaya yardımcı olur, böylece projenin başarıyla yönetilmesine katkıda bulunur.

Fizibilite çalışması aşağıda belirtilen altı maddeyi içerecektir:

1- Proje Kapsamı: İş sorununun tanımı veya ele alınması gereken fırsattır. Açık, kısa ve doğru olmalıdır.

2- Mevcut Analiz: Sistemin/yazılım ürününün vb. bir anlayışın tanımlanması ve oluşturulmasıdır. Bu analize dayanarak, mevcut sistem veya yazılım ürününün doğru çalıştığı, bazı küçük değişikliklere ihtiyaç olduğu veya tam bir yükselme/değiştirme gerektiği belirlenebilir.

3- Gereksinimler: Paydaşların ihtiyaçlarına ve kısıtlamalarına ilişkin proje gereksinimlerinin tanımlanmasıdır. Yazılım ve sistem için gereksinimler farklı şekillerde belirlenmektedir. Örneğin;

- İş, sözleşme ve mevzuat süreçleri,

- Son kullanıcı fonksiyonel ihtiyaçları,

- Operasyonel ve mühendislik parametrelerini tanımlayan teknik ve fiziksel öz nitelikler.

4- Yaklaşım: Önerilen sistem/yazılım çözümü için gereksinimleri karşılamak üzere eylem planının tanımlanmasıdır. Bu adım, dikkate alınan alternatifleri ve tercih edilen çözümün seçilme gereksinimini açıkça tanımlamaktadır. Bu, mevcut yapıların ve ticari alternatiflerin kullanımının dikkate alındığı süreçtir.

5- Değerlendirme: Fizibilite çalışmasında önceden tanımlanmış öğelere dayanarak, projenin maliyet etkinliğinin incelenmesidir. Nihai rapor, seçilen yaklaşımın maliyet etkinliğini ele almaktadır. Nihai raporun öğeleri şunları içerir:

- Tercih edilen çözüm seçilirse, projenin tahmini toplam maliyeti ve aşağıdakiler dahil bir maliyet karşılaştırması sağlama alternatifleri:

- Tamamlanması için gerekli çalışan saatleri,

- Materyal ve olanak maliyetleri,

- Tedarikçi ve üçüncü taraf yüklenici maliyetleri,

- Proje zaman programı başlangıç ve bitiş tarihleri

- Maliyet-fayda analizini, yatırım getirisini vb. kapsayan bir maliyet değerlendirme özeti.

6- İnceleme: Fizibilite çalışmasının daha önce tanımlanmış öğelerinin hem fizibilite çalışmasının eksiksizliğini ve güvenilirliğini doğrulamak hem de nihai kararı vermeden önce projeyi onaylama/reddetme/düzeltme/isteme kararı vermek için gözden geçirilmesidir. İnceleme ve rapor kilit paydaşlarla birlikte yapılmaktadır. Fizibilite çalışmasının reddedilmesinin gerekçesi açıklanmalı ve gelecekteki proje çalışmalarında kullanılmak üzere derse alınanların listesinin parçası olarak belgeye eklenmelidir.

Fizibilite çalışması sırasında, bilgi sistemleri denetçisi aşağıdakileri yapmalıdır:

- Uygun olduğundan emin olmak için faz belgelerinin gözden geçirilmesi,

- Tüm maliyet gerekçelerinin/faydalarının doğrulanabilir olup olmadığının ve beklenen maliyetleri ve beklenen faydaları gösterip göstermediklerinin belirlenmesi,

- İhtiyacın kritikliğinin belirlenmesi ve tespit edilmesi,

- Hali hazırda mevcut olan sistemlerle bir çözüm elde edilip edilmeyeceğinin belirlenmesi, çözüme ulaşılamamışsa, alternatif çözümlerin makul olup olmadığına ilişkin değerlendirmenin gözden geçirilmesi,

- Seçilen çözümün uygunluğunun tespit edilmesi.

Fizibilite çalışmasında ihtiyaç tanımına ilişkin uygulanması sebebiyle bilgi sistemleri denetçisi aşağıdaki işlevleri yerine getirmelidir:

- Ayrıntılı gereksinimler tanımı belgesinin edinilmesi ve ilgili bölümler ile görüşmeler yaparak doğruluğunun ve tamlığının incelenmesi,

- Proje ekibindeki kilit üyelerin belirlenmesi ve etkilenen tüm kullanıcı gruplarının uygun temsile sahip olduğunun doğrulanması,

- Projenin maliyeti ve başlatılmasına ilişkin onayların alındığının teyit edilmesi,

- Kullanıcının gereksinimlerini karşıladığından emin olmak için kavramsal tasarım spesifikasyonlarının incelenmesi,

- Kontrol spesifikasyonlarının tanımlandığından emin olmak için kavramsal tasarımın incelenmesi,

- Makul sayıda tedarikçiden proje kapsamını ve kullanıcı gereksinimlerini kapsayan teklifin alınıp alınmadığının incelenmesi,

- Kullanıcı kabul testi spesifikasyonlarının incelenmesi,

- Uygulamanın gömülü bir denetim rutini kullanımına aday olup olmadığının belirlenmesi, eğer adaysa rutinin sistemin kavramsal tasarımına dahil edilmesinin istenmesi.

1.1.2.2. Sistem Geliştirme Projesi Maliyet Tahminleri

Bilgi sistemi projelerinde maliyet tahmini için kullanılan dört farklı yöntem bulunmaktadır. Bunlar:

1- Benzer Tahmin: Benzer tahmin, proje yöneticisinin geçmiş projelerine yönelik tahminleri ile maliyetin planlanmasıdır. En hızlı yöntem olarak değerlendirilmektedir. Bu yöntemde, projenin maliyet tahmini, benzer önceki projelerin deneyimlerine dayanılarak yapılır. Proje yöneticisi veya ekibi, geçmiş projelerin maliyetlerini inceleyerek, yeni projenin tahmini maliyetini bu referans projelere dayalı olarak hesaplar. Bu yöntem hızlıdır, ancak tahminler, benzer projelerin ne kadar iyi bir öngörüleyici olduğuna bağlı olarak doğruluk seviyesinde değişiklik gösterebilir.

2- Parametrik Tahmin: Parametrik tahmin, proje yöneticisinin benzer tahminlerde kullanılan geçmiş veri analizine dayalı olarak gerçekleştirdiği planlamadır. Bu yaklaşım benzer tahminden daha doğru olarak değerlendirilmektedir. Parametrik tahmin, projenin boyutunu veya karmaşıklığını temsil eden parametrelerin kullanılmasıyla gerçekleştirilir. Geçmiş projelerin verileri kullanılarak, belirli bir

birim maliyeti veya parametrik modelleme yöntemleriyle yeni projenin tahmini maliyeti hesaplanır. Bu yaklaşım, benzer tahminden daha spesifik ve doğru tahminler sağlayabilir.

3- Aşağıdan Yukarıya Tahmin: Bu yöntemde, projedeki her bir faaliyetin maliyeti en büyük ayrıntıya kadar tahminlenir ve maliyete eklenmektedir. En çok zaman alan bu yaklaşım en doğru tahmin yöntemidir. Bu yöntemde, projenin her bir bileşeninin veya faaliyetinin ayrı ayrı tahmini yapılır ve sonunda bu tahminler toplanarak toplam proje maliyeti elde edilir. Her bir faaliyetin ayrıntılı bir tahmini gerektirdiği için zaman alıcıdır, ancak en doğru tahmin sonuçlarını sağlayabilir. Genellikle WBS (Work Breakdown Structure - İş Bölüm Yapısı) kullanılarak uygulanır.

4- Gerçek Maliyet: Gerçek maliyet, geçmiş projeler sırasında aynı sisteme ilişkin gerçekleşen gerçek maliyetlerin değerlendirilmesi yöntemidir. Gerçek maliyet tahmininde, geçmiş projeler sırasında aynı veya benzer bir sistemin inşa edilmesi sırasında ortaya çıkan gerçek maliyetlerin analizi yapılır. Gerçekleşmiş maliyet verileri, yeni projenin tahmini maliyetinin hesaplanmasına yardımcı olur. Bu yöntem, geçmiş projelerin benzerlik derecesine bağlı olarak güvenilir olabilir.

Proje maliyetlerinin tahminine ilişkin olarak, aşağıda da açıklamalarına yer verilen, yazılım boyutu tahminleri, fonksiyon nokta analizi ve yazılım maliyet tahminlerine ilişkin çalışmalar yapılır. Proje maliyet tahmini, yazılım geliştirme projelerinin başarıyla tamamlanması için önemli bir adımdır. Tahminlerin ne kadar doğru olduğu, projenin başarı şansını ve bütçe yönetimini etkileyebilir. Bu nedenle, projenin özelliklerine, gereksinimlerine ve geçmiş deneyimlere dayalı olarak en uygun tahmin yöntemi seçilmelidir. Ayrıca, maliyet tahminlerinin projenin her aşamasında güncellenmesi ve izlenmesi de gereklidir.

a. Yazılım Boyutu Tahminleri

Yazılım boyutu tahmini, bir yazılım ürünü geliştirmek için gereken çabayı belirlemek için önemli bir özelliktir. Yetersiz, sorgulanabilir ve usulsüz veriler ışığında kalkınma görevlerini oluşturulması veya sürdürülmesi için gerekli olan en pratik efor ölçüsünün (bireysel saat ya da sermaye olarak iletilen) öngörülmesini sağlayan metodolojidir. Yazılım Çaba Tahminleri (Software Effort Estimation-SEE), yazılımı geliştirmek veya sürdürmek için gereken çabanın en makul şekilde kullanımını öngören prosedürdür.

Yazılım boyutlandırma veya yazılım boyutu tahmini, diğer yazılım proje yönetimi faaliyetlerinin (tahmin etme veya izleme gibi) uygulayabilmesi için bir yazılım uygulaması veya bileşeninin boyutunun belirlenmesi veya tahmin edilmesi için kullanılan yazılım mühendisliğindeki bir faaliyettir. Boyut, tıpkı ağırlığın somut bir malzemenin doğal özelliği olması gibi, bir yazılım parçasının doğal bir özelliğidir.

Geliştirilecek uygulama yazılımının göreceli fiziksel büyüklüğünü belirlenmesini sağlamaktadır. Tahminler, kaynakların tahsisine rehberlik etmesi, geliştirme için gereken zaman ve maliyetin değerlendirilmesi ve kaynakların gerektirdiği eforun karşılaştırılması için kullanılabilmektedir.

Yazılım boyutu tahmininin önemi aşağıdaki maddeler bağlamında ele alınmaktadır:

Kaynak Tahsisi: Yazılım boyutu tahminleri, projenin ne kadar büyük olduğunu ve ne tür kaynaklara ihtiyaç duyulacağını belirlemek için kullanılır. Bu, işgücü, zaman ve mali kaynakların daha verimli bir şekilde tahsis edilmesini sağlar.

Zaman ve Bütçe Yönetimi: Doğru bir yazılım boyutu tahmini, projenin ne kadar süreceğini ve ne kadar maliyet gerektireceğini daha iyi anlamanıza yardımcı olur. Bu, proje yöneticilerinin süreci daha etkili bir şekilde planlamalarına ve takip etmelerine olanak tanır.

Risk Yönetimi: Yanlış veya eksik bir boyut tahmini, projenin başarısız olma riskini artırabilir. Doğru bir tahmin, beklenmeyen sorunların önlenmesine yardımcı olur.

İzleme ve Değerlendirme: Projeyi ilerledikçe, tahminlerle gerçekleşen arasındaki farkları izlemek, proje sağlığı hakkında bilgi sağlar ve gerektiğinde ayarlamalar yapılmasına yardımcı olur.

Veri Tabanlı Kararlar: Yazılım boyutu tahminleri, projenin gerektirdiği kaynakların ve çabanın somut bir şekilde anlaşılmasına katkıda bulunur. Bu veriler, organizasyonun gelecekteki projelerini daha iyi planlamasına olanak tanır.

Yazılım boyutu tahminleri, proje yönetiminde başarılı bir yazılım geliştirme süreci için kritik bir adımdır. Bu tahminler, projenin başlangıcında kaynak tahsisi, zaman planlaması ve maliyet değerlendirmeleri yapmak için kullanılır. Boyut tahmininin doğru yapılması, kaynakların verimli kullanılmasını sağlar ve projede beklenmeyen değişikliklerin önüne geçer. Modern yazılım geliştirme süreçlerinde, yazılım boyutu tahminlerinin daha hassas yapılabilmesi için birçok yöntem ve teknoloji kullanılmaktadır.

Birincil olarak kullanılan yazılım boyutu tahmin yöntemleri arasında fonksiyonel nokta analizi (Function Point Analysis - FPA), satır sayısı (Lines of Code - LOC) ve çaba tahmini metodolojileri yer alır. Fonksiyonel nokta analizi, yazılımın işlevsel gereksinimlerini ölçen bir tekniktir ve projenin gerektirdiği işlevlerin karmaşıklığını dikkate alarak tahmin yapar. Satır sayısı (LOC), yazılımın kod uzunluğunu belirleyerek boyut tahmini yapar, ancak modern yazılım geliştirme yaklaşımlarında bu yöntem, yazılımın doğasına göre sınırlı kullanılır. Çaba tahmini metodolojileri ise yazılımın gerektirdiği toplam iş gücü ile ilgili tahminler yapar ve genellikle karmaşık projelerde daha doğru sonuçlar verir.

Modern Yazılım Boyutu Tahmin Araçları: Günümüzde yazılım boyutu tahminlerini daha doğru ve verimli yapmak için yapay zeka ve makine öğrenimi gibi teknolojiler kullanılmaktadır. Bu araçlar, geçmiş proje verilerini analiz ederek yeni projelerin tahmin edilmesine yardımcı olabilir. Örneğin, makine öğrenimi algoritmaları, geçmiş projelerin verilerinden öğrenerek benzer projelerin gereksinimlerini tahmin etmekte kullanılabilir. Bu, geleneksel yöntemlere göre daha hızlı ve daha hassas tahminler yapılmasına olanak tanır.

Ayrıca, yazılım boyutu tahminlerinde çevik (agile) geliştirme metodolojileri de önemli bir yer tutar. Agile projelerinde, yazılım boyutu tahminleri genellikle kullanıcı hikayesi puanı (user story point) ve sprint tahminleri gibi tekniklerle yapılır. Bu, yazılımın fonksiyonel gereksinimlerini küçük, yönetilebilir parçalara ayırarak tahmin yapmayı sağlar. Bu yöntem, projenin ilerleyişine göre tahminlerin sürekli olarak gözden geçirilmesini ve güncellenmesini mümkün kılar.

Yazılım Boyutu Tahmininin Zorlukları ve Riskler: Yazılım boyutu tahmininin doğru yapılmaması, projede maliyet aşımlarına, zaman kaybına ve başarısızlığa yol açabilir. Yanlış tahminler, projenin kaynaklarını aşırı kullanmaya veya yeterli kaynağa sahip olmamaya neden olabilir. Ayrıca, yazılım boyutu tahmininin sürekli olarak gözden geçirilmesi ve güncellenmesi önemlidir, çünkü proje ilerledikçe bazı tahminler gerçeği yansıtmayabilir. Bununla birlikte, tahminlerin yapılması sırasında dikkat edilmesi gereken bir diğer önemli husus da takımın deneyimi ve projede kullanılan teknolojilerin karmaşıklığıdır. Her yeni teknolojinin tahminleri etkileyebileceği için, bu faktörler de göz önünde bulundurulmalıdır.

Sonuç olarak, yazılım boyutu tahminleri, bir yazılım geliştirme projesinin başarısı için kritik bir adımdır. Doğru tahminler, kaynakların etkin bir şekilde kullanılmasını ve projenin hedeflerine ulaşılmasını sağlar. Ancak, tahminlerin doğruluğu, kullanılan yöntemler ve araçlarla doğrudan ilişkilidir. Modern araçlar ve teknolojiler, yazılım boyutu tahminlerini daha hassas hale getirerek projelerin daha verimli bir şekilde yönetilmesine yardımcı olmaktadır.

b. Fonksiyon Nokta Analizi

Büyük iş uygulamalarının geliştirilmesinde karmaşıklığı tahmin etmek için kullanılan çok noktalı bir tekniktir. Bu analiz bir bilgi sistemi boyutunun, kullanıcının etkileşimde bulunduğu girdi, çıktı, arayüz, dosya ve sorguların sayı ve karmaşıklığına bağlı ölçüsüdür. Bu doğrudan boyut odaklı ölçümlere karşı yazılım boyutunun ve geliştirme sürecinin dolaylı ölçüsüdür. Fonksiyon noktaları ilk önce belirli bir girişin basit, ortalama veya karmaşık olup olmadığının belirlenmesi için bir tablo tanımlanarak hesaplanmaktadır. Kullanıcı girişi, kullanıcı çıktı, kullanıcı sorgu, dosya ve harici arayüz sayıları dahil olmak üzere beş fonksiyon noktası sayım değeri tanımlanmaktadır.

Tablo girişlerinin tanımlanmasının ardından, fonksiyon noktasının türetilmesindeki toplam sayı güvenilirlik, kritiklik, karmaşıklık, tekrar kullanılabilirlik, değiştirilebilirlik ve taşınabilirlik gibi konulara ilişkin sorulara verilen yanıtlara dayalı olarak karmaşıklık ayarlama değerlerini

(derecelendirme faktörleri) dikkate alan bir algoritma ile hesaplanmaktadır. Bu denklemden türetilen fonksiyon noktaları daha sonra maliyet, program, üretkenlik ve kalite kriterleri için bir ölçüt olarak Source Line of Code (Kaynak Kod Satır Sayısı-SLOC) sayılarına benzer şekilde kullanılmaktadır.

Fonksiyon nokta analizi, iş uygulamalarının tahminlenmesinde oldukça başarılı bir yöntemdir. Ancak diğer yazılım türleri (işletim sistemi, süreç kontrolü, iletişim ve mühendislik vb.) için yeterli değildir. Diğer tahmin yöntemleri bu tür yazılımlar için daha uygundur ve Constructive Cost Model (Yapıcı Maliyet Modeli-COCOMO), De Marco ve Watson-Felix'in fonksiyon nokta analizi özelliklerini içermektedir.

Fonksiyon Nokta Analizi Ölçütlerinin Hesaplanması					
Ölçüm Parametresi	**Sayı**	**Ağırlık Faktörü**			
		Basit	**Ortalama**	**Karmaşık**	**Sonuçlar**
Kullanıcı girdi sayısı		3	4	6	…..
Kullanıcı çıktı sayısı		4	5	7	…..
Kullanıcı sorgu sayısı		3	4	6	…..
Dosya sayısı		7	10	15	…..
Harici arayüz sayısı		5	7	10	…..
Toplam sayı	FP yöntemlerini kullanan kuruluşlar, belirli bir girişin basit, ortalama veya karmaşık olup olmadığını belirlemek için kriterler geliştirir.				

Fonksiyon nokta analizi yöntemin önemine aşağıda maddeler halinde değinilmiştir:

- Karmaşıklığı Tahmin Etmek İçin Kullanılır: Fonksiyon nokta analizi, bir bilgi sistemi veya yazılımın karmaşıklığını tahmin etmek için kullanılır. Bu karmaşıklık, kullanıcıların sistemi nasıl kullanacaklarına, hangi girdileri ve çıktıları olacağına, dosyaların nasıl yönetileceğine ve sorguların ne kadar karmaşık olduğuna bağlıdır.

- Dolaylı Bir Boyut Ölçüsüdür: Fonksiyon nokta analizi, yazılım boyutunu doğrudan satır kodu (SLOC) gibi ölçülerle değil, kullanıcıların ihtiyaçlarına ve sistemin işlevselliğine dayalı olarak tahmin eder. Bu, yazılımın gerçek işlevselliğine odaklanmasını sağlar.

- Fonksiyon Noktalarının Hesaplanması: Fonksiyon noktaları, kullanıcı girişi, kullanıcı çıktısı, kullanıcı sorgusu, dosya ve harici arayüz sayıları gibi özel bir tablo kullanılarak hesaplanır. Bu tabloya göre, her bir işlevin karmaşıklığı basit, ortalama veya karmaşık olarak sınıflandırılır ve puanlarla ölçülür.

- Karmaşıklık Ayarlama Değerleri: Fonksiyon noktalarının hesaplanmasının ardından, güvenilirlik, kritiklik, karmaşıklık, tekrar kullanılabilirlik, değiştirilebilirlik ve taşınabilirlik gibi faktörlere dayalı olarak karmaşıklık ayarlama değerleri eklenir. Bu, projenin özgün gereksinimlerini ve zorluklarını yansıtır.

- Maliyet ve Üretkenlik Tahmini: Fonksiyon noktaları, maliyet, program, üretkenlik ve kalite kriterleri için bir ölçüt olarak kullanılır. Örneğin, yazılımın geliştirme süresini ve maliyetini tahmin etmek için kullanılabilir.

- İş Uygulamaları İçin Başarılı Bir Tahmin Yöntemi: Fonksiyon nokta analizi, iş uygulamalarının tahminlenmesinde oldukça başarılı bir yöntem olarak kabul edilir. Bu yöntem, proje yöneticilerine ve yazılım geliştiricilere işleri planlama ve kaynakları yönetme konusunda yardımcı olur.

- Diğer Yazılım Türleri İçin Uygun Değil: Fonksiyon nokta analizi, iş uygulamaları için ideal bir tahmin yöntemi olsa da, diğer yazılım türleri (örneğin, işletim sistemleri, süreç kontrolü, iletişim yazılımları vb.) için uygun değildir. Bu tür yazılımlar için daha farklı tahmin yöntemleri kullanılması gerekebilir.

- Diğer Tahmin Yöntemleri: Fonksiyon nokta analizi dışında, yazılım projeleri için tahmin yöntemleri arasında Yapıcı Maliyet Modeli (Constructive Cost Model - COCOMO) gibi yöntemler de bulunmaktadır. Projenin doğası ve gereksinimlerine bağlı olarak farklı tahmin yöntemleri tercih edilebilir.

c. Yazılım Maliyeti Tahminleri

Yazılım maliyeti, yazılım boyutu tahmininin sonucudur. Yazılımın gerçekleştirilmesi için gereken bütçe tahmin edilir. Böylece proje kapsamının en uygun şekilde belirlenmesi ve projedeki belirsizliklerin azaltılması sağlanır Bilgi sistemi geliştirme projelerinin her aşamasında maliyet tahmini yapılır. Bu tahminleme için kullanılan otomatik teknikler bulunmaktadır. Söz konusu otomatik maliyet tahmini tekniklerinin kullanılabilmesi için bilgi sistemi genellikle ana bileşenlerine ayrılmakta ve bir dizi maliyet sürücüsü oluşturulmaktadır. Ayrılan bileşenlerin içerikleri aşağıda belirtildiği gibidir:

Yazılım Maliyeti ve Tahminleri:

Yazılım maliyeti, bir projenin boyutu, karmaşıklığı ve kapsamı gibi faktörlere dayalı olarak tahmin edilir. Bu tahminler, projenin başarılı bir şekilde planlanması, yönetilmesi ve bütçenin kontrol edilmesi için kritik bir öneme sahiptir.

Yazılım maliyeti tahminleri, projelerin gereksinimlerini doğru bir şekilde belirlemek, kaynakları etkin bir şekilde tahsis etmek ve maliyet artışlarını engellemek için kritik bir araçtır. Modern yazılım projelerinde, maliyet tahminleri sadece teknik bileşenlere dayanmaz, aynı zamanda yazılımın geliştirilmesinde kullanılan metodolojiler, sistem geliştirme yaşam döngüsü (SDLC) aşamaları ve kullanılan teknolojilerin karmaşıklığı da göz önünde bulundurulur. Özellikle agile metodolojiler (çevik yazılım geliştirme) kullanıldığında, proje süresince sürekli olarak maliyet tahminlerinin güncellenmesi gerekebilir. Bu, yazılımın gelişen ihtiyaçlara ve değişen gereksinimlere adapte olmasına olanak tanır.

Proje Kapsamının Belirlenmesi:

Yazılım maliyet tahminleri, proje kapsamının daha iyi belirlenmesine yardımcı olur. Bu, projenin hedeflerinin, gereksinimlerinin ve teslimatların net bir şekilde tanımlanması anlamına gelir. Doğru bir maliyet tahmini, projenin bütçesini ve süresini etkileyen belirleyici bir faktördür.

Modern yazılım geliştirme süreçlerinde, proje kapsamı belirlenirken sadece işlevsel gereksinimler değil, aynı zamanda yazılımın sürdürülebilirliği, güvenliği ve entegrasyonu gibi gereksinimler de göz önüne alınmalıdır. Özellikle bulut tabanlı yazılım çözümleri ve microservices (mikro hizmetler) mimarisi gibi modern teknolojiler, proje kapsamını doğrudan etkileyebilir, çünkü bu tür yapılar genellikle daha fazla entegrasyon, güvenlik önlemleri ve veritabanı yönetimi gerektirir.

Otomatik Maliyet Tahmini Teknikleri:

Yazılım geliştirme maliyetlerini tahmin etmek için otomatik teknikler kullanılır. Bu teknikler, geçmiş projelerin verileri, iş yükü analizi, karmaşıklık faktörleri ve diğer parametreleri kullanarak maliyet tahminlerini oluşturur. Bu sayede daha doğru ve güvenilir tahminler elde edilir.

Makine öğrenimi (ML) ve yapay zeka (AI), yazılım geliştirme maliyet tahminlerinde giderek daha fazla kullanılmaktadır. Bu teknolojiler, daha önceki projelerin verilerini analiz ederek yeni projelerin gereksinimlerini tahmin etmekte oldukça başarılıdır. AI destekli araçlar, tahminlerin doğruluğunu artırırken, zaman ve kaynak yönetimi konularında daha hızlı kararlar alınmasını sağlar. Ayrıca, neural network (sinir ağları) gibi derin öğrenme algoritmalarının kullanılması, daha karmaşık yazılım projeleri için maliyet tahminlerini daha doğru hale getirebilir.

Bilgi Sisteminin Bileşenlere Ayrılması:

Yazılım projeleri, genellikle ana bileşenlere ayrılır. Bu bileşenler, projenin farklı işlevselliğini veya modüllerini temsil eder. Her bir bileşen, maliyet tahminlerinin daha ayrıntılı ve hassas bir şekilde yapılmasına olanak tanır.

Modern yazılım projelerinde modüler tasarım (modular design) ve CI/CD (Continuous Integration / Continuous Deployment) süreçleri de bu bileşenleri daha ayrıntılı bir şekilde ele almayı gerektirir. Modüler yazılım geliştirme, her bir modülün bağımsız olarak geliştirilip test edilmesine olanak tanır ve bu da maliyet tahminlerinin daha hassas yapılabilmesini sağlar. Ayrıca, CI/CD süreçleri, yazılımın sürekli entegrasyonunu ve dağıtımını sağlarken, yazılımın her aşamasının maliyetini izlemeyi kolaylaştırır.

Ana Bileşenler ve Maliyet Sürücüleri:

Kaynak Kod Dili: Hangi programlama dilinin kullanılacağı, yazılımın geliştirme süresi ve maliyeti üzerinde önemli bir etkiye sahiptir. Farklı diller, farklı özelliklere ve karmaşıklıklara sahiptir.

Yürütme Süresi Kısaltmaları: Yazılımın hızlı çalışması için yapılan iyileştirmeler maliyeti artırabilir. Örneğin, daha hızlı bir donanım gereksinimi veya özel optimizasyonlar gerekebilir.

Ana Depolama Kısaltmaları: Ana depolama alanının boyutu ve türü, verilerin depolanması ve erişimi için önemlidir.

Veri Depolama Kısaltmaları: Verilerin nasıl depolandığı, yedeklendiği ve yönetildiği maliyeti etkiler.

Bilgisayar Erişimi: Hangi tür bilgisayarların kullanılacağı, donanım maliyetini belirler.

Geliştirme İçin Kullanılan Hedef Makine: Geliştirme, test ve dağıtım için kullanılacak olan donanım veya sanal makinelerin maliyeti önemlidir.

Güvenlik Ortamı: Proje güvenliği için gereken yazılım ve donanım, maliyeti artırabilir.

Personel Deneyimi: Proje ekibinin deneyimi, iş gücü maliyetlerini etkiler. Daha deneyimli personel genellikle daha yüksek maaş talep edebilir.

Yazılım projelerinde maliyetleri etkileyen ana bileşenler arasında kullanılan programlama dili, yürütme süresi kısaltmaları, veri depolama yöntemleri ve kullanılan güvenlik teknolojileri yer alır. Bu bileşenlerin her biri, yazılımın geliştirilmesindeki çabayı ve maliyeti doğrudan etkiler. Günümüzde bulut bilişim (cloud computing) ve sunucusuz mimari (serverless architecture) gibi teknolojiler, yazılımın geliştirilmesi ve dağıtım süreçlerini daha verimli hale getirmektedir. Bu teknolojiler, fiziksel donanım gereksinimlerini ortadan kaldırarak, yazılım geliştirme maliyetlerini önemli ölçüde azaltabilir. Ayrıca, blockchain teknolojisi ve şifreleme algoritmaları (encryption algorithms) gibi güvenlik teknolojileri, yazılım geliştirme süreçlerinde daha fazla zaman ve kaynak gerektirebilir, dolayısıyla maliyetleri artırabilir.

Maliyet Tahminlerinin Geliştirilmesi:

Tüm bu maliyet sürücüleri ve bileşenler belirlendikten sonra, yazılım projesinin ve bilgi sisteminin maliyet tahminleri daha ayrıntılı ve kesin bir şekilde hesaplanabilir. Bu tahminler, proje yöneticilerine bütçe tahsisi ve kaynak yönetimi konularında rehberlik eder. Tüm sürücüler belirlendikten sonra program bilgi sistemi ve toplam projenin maliyet tahminlerini geliştirecektir.

Modern yazılım geliştirme süreçlerinde, çevik (agile) metodolojiler ve Scrum çerçevesi (Scrum framework) gibi yöntemler, projenin maliyet tahminlerini daha esnek ve dinamik bir şekilde yapabilmeyi mümkün kılar. Agile projelerinde, yazılım maliyeti tahminleri sürekli olarak güncellenebilir ve proje sürecinde değişen gereksinimlere göre revize edilebilir. Bu da maliyet yönetimini daha doğru ve esnek hale getirir. Ayrıca, yazılımın yaşam döngüsü boyunca yapılan her değişiklik, maliyet tahminlerini etkileyebilir, bu nedenle değişiklik yönetimi süreçlerinin de yazılım maliyeti tahminlerinin bir parçası olması gerekmektedir.

1.1.2.3. Zaman Çerçevesinin Oluşturulması ve Çizelgelenmesi

Projeler için zaman çerçevesinin belirli bir çizelge kapsamında oluşturulması, etkinlik ve verimliliği artırma amacıyla belirli faaliyetler üzerinde harcanacak sürenin kontrol edilmesi sağlar. Zaman yönetimi için çeşitli araç ve yöntemler kullanılır. Bu araçlardan biri olan bütçeleme, her göreve dahil olan insan ve makine eforunun toplamı olup, zamanlama ve görevler arasındaki sıralı ilişkidir. Bu görevlerin belirlenmesini sağlayan iki öğeye göre düzenlenir. Bunlar:

- Görevler arası mantıksal sıralı ilişkiye dayalı ve görevlerin paralel yürütülmesi halinde en erken başlangıç tarihi,

- Çalışan dahil tüm kaynakların uygunluğuna göre (izin/tatil süreçleri, işe alım süresi vb. dahil edilerek) en son beklenen bitiş tarihi.

Program, aşağıda da belirtilen teknikler (Gantt şeması, Kritik Yol Metodu (Critical Path Method -CPM), Program Değerlendirme ve Kontrol Tekniği (Project Evaluation Review Technique-PERT) vb.)

kullanılarak grafiksel olarak da gösterilebilir. Proje sürecince durumun analiz edilmesi önemlidir. Kilit noktalar/ kritik faaliyetler ve kilometre taşları analiz edilerek program ve bütçedeki sapmalar analiz edilmektedir. Varyanslar ve varyans analizi yönetime zamanında bildirilmelidir.

Zaman çerçevesi oluşturulurken, her projenin etkinliğini ve verimliliğini artırmak için farklı teknikler ve araçlar kullanılır. Proje yöneticilerinin zaman çerçevesini oluştururken dikkate alması gereken birkaç önemli faktör bulunmaktadır:

Görevlerin Sıralanması ve Önceliklendirilmesi

Projelerde görevler arasındaki ilişkiyi ve önceliği belirlemek, projenin doğru şekilde ilerlemesini sağlamak için kritiktir. Bu süreçte, zaman çerçevesinin belirlenmesinde Kritik Yol Metodu (Critical Path Method - CPM) ve Program Değerlendirme ve Kontrol Tekniği (PERT) gibi zaman yönetimi teknikleri kullanılır.

Kritik Yol Metodu (CPM): Projelerde görevlerin birbirine bağlı olduğu ve her görevin belirli bir sürede tamamlanması gereken projelerde, görevlerin sıralı ilişkileri ve kritik yol belirlenir. Bu, projede yapılan her görev arasındaki süreyi minimize etmek için yardımcı olur.

PERT Yöntemi: Projelerde belirsizlik durumunu yönetmek için kullanılır. Özellikle, görevlerin tamamlanma süresinin kesin olmaması durumunda, PERT daha etkili sonuçlar verir ve projedeki belirsizliği hesaplar.

Kaynakların Tahsisi ve Yönetimi

Zaman çerçevesinin oluşturulmasında, her göreve ait kaynakların uygun şekilde tahsis edilmesi gerekir. Bu, hem insan hem de fiziksel kaynakları içerir. Bu aşamada dikkat edilmesi gereken başlıca unsurlar şunlardır:

İnsan Kaynakları Yönetimi: Her görevin yürütülmesi için gerekli olan personel, yeteneklerine ve projedeki sorumluluklarına göre belirlenmelidir. Ayrıca, tatiller, izinler ve iş gücü dağılımı gibi faktörler de göz önünde bulundurulmalıdır.

Fiziksel Kaynaklar Yönetimi: Projede kullanılan ekipman, yazılım ve diğer altyapı unsurları da zaman çizelgesine entegre edilmelidir. Bu kaynakların temin edilmesi, dağıtılması ve kullanılması süreçleri etkili bir şekilde izlenmelidir.

Çevik Yöntemler (Agile): Zaman çerçevesinin oluşturulmasında çevik (Agile) proje yönetim metodolojileri de kullanılabilir. Çevik yönetim, kaynakların esnek bir şekilde tahsis edilmesini sağlar ve projelerin dinamiklerine göre hızla uyum sağlamayı kolaylaştırır.

Zaman Çizelgesinin İzlenmesi ve Güncellenmesi

Zaman çerçevesinin oluşturulmasının ardından, projenin ilerleyişi takip edilmelidir. Zaman çizelgesindeki sapmalar, başta bütçe ve kaynaklar olmak üzere projede önemli aksaklıklara yol açabilir. Bu nedenle, zaman çizelgesinin düzenli olarak izlenmesi ve güncellenmesi gereklidir.

Zaman Çizelgesi İzleme Araçları: Çeşitli dijital araçlar (örneğin, Microsoft Project, Asana, Jira, Trello) zaman çizelgesinin izlenmesinde kullanılabilir. Bu araçlar, proje ekibine gerçek zamanlı güncellemeler sağlar ve projede sapmalar olduğunda hızlıca müdahale etmeyi mümkün kılar.

Varyans Analizi: Varyans analizi, planlanan zaman çizelgesi ile gerçekte gerçekleşen zaman arasındaki farkları belirler ve proje yöneticisinin sapmaları yönetmesine yardımcı olur. Bu analiz, projenin takibinin yapıldığı ve düzeltici önlemlerin alındığı önemli bir süreçtir.

Kilometre Taşları ve Kritik Faaliyetlerin Belirlenmesi

Projenin önemli dönüm noktaları ve kritik faaliyetlerinin belirlenmesi, proje yöneticisinin bu noktalar üzerinde yoğunlaşarak proje ilerleyişini denetlemesine olanak tanır.

Kilometre Taşları: Proje sürecinde önemli adımların tanımlanması ve her bir kilometre taşının başarılmasına yönelik hedefler belirlenmesi gereklidir. Bu, projede yöneticilerin ve ekiplerin net bir yol haritasına sahip olmalarını sağlar.

Kritik Faaliyetler: Kritik faaliyetler, projenin zaman çizelgesinde yer alan ve tamamlanması gereken en önemli adımlardır. Bu faaliyetlerin zamanında tamamlanması, projenin genel başarısını doğrudan etkiler. Kritik faaliyetlerin doğru bir şekilde tanımlanması ve izlenmesi, projenin doğru yolda ilerlemesini sağlar.

Zaman Çizelgesinin Risk Yönetimi ile Entegre Edilmesi

Projelerdeki zaman çerçevesi, yalnızca görevler ve kaynaklar ile değil, aynı zamanda olası riskler ile de entegre edilmelidir. Proje sürecinde karşılaşılabilecek riskler, zaman çizelgesinde büyük sapmalara yol açabilir.

Risklerin Zaman Çizelgesine Etkisi: Projelerde öngörülemeyen durumlar, zaman çizelgesinin ciddi şekilde sapmasına neden olabilir. Bu nedenle, risklerin zaman çizelgesine etkisi belirlenmeli ve olası gecikmelere karşı esnek planlar yapılmalıdır.

Proaktif Risk Yönetimi: Proaktif olarak riskleri izlemek ve önceden belirlemek, zaman çizelgesinin sağlıklı bir şekilde yönetilmesine yardımcı olur. Zaman çizelgesi yönetiminde risklerin yönetilmesi, özellikle büyük projelerde kritik öneme sahiptir.

1.1.2.4. Gantt Şemaları

Gantt şemaları Henry Gantt tarafından tasarlanan, iş yönetiminde planlılığını sağlamaya yönelik grafik tasarımıdır. Gantt şemaları, bir projenin tamamlanması için gerekli aktivitelerin planlanmasını desteklemektedir. Projelerin planlanması, projenin zamanlamasını ayarlanması, kaynakların belirlenmesi ve projenin yönetimini sağlanması amacıyla tasarlanan Gantt şeması proje yöneticileri için kurtarıcı bir diyagramdır. Büyük ve karmaşık projelerin sorunsuz bir şekilde ilerlemesine yardımcı olan etkili bir proje yönetim aracıdır. Sadece belirli bir proje zaman çizelgesini görselleştirmekle kalmaz, aynı zamanda ekipler ve paydaşlar arasında anlayış oluşturur ve beklentileri belirler.

Gantt şeması bir bakışta, bireysel faaliyetlerin ne olduğunu, hangi aşamada kim tarafından yapılacağını, bu faaliyetlerin her birinin ne zaman başlayıp bitmesi gerektiğini, planlanan zaman ve geçen fiili zamanın karşılaştırılması ve buna göre aksiyon alınması gibi durumların planlanmasını, faaliyetler arasında herhangi bir çakışma olup olmadığını ve projenin belirli bir süre içinde nasıl tamamlanabileceğini göstermektedir. Aynı zamanda hangi aktivitelerin eş zamanlı ve hangi sırayla yürütüleceğini de göstermektedir. Gantt şeması, bir projede ihtiyaç duyulan tüm özelliklerin tek bir çizelgede gösterilmesini sağlar. Bu özellikler aşağıdaki gibidir:

- Kilometre Taşları (Milestones): Projeyi bir aşamadan başka bir aşamaya taşıyan olay, tarih, karar veya teslimlerdir.

- İş Paketleri (Work Package): Birbiriyle alakalı yapılacak işler bütünüdür. Herhangi bir eylem barındırmaz.

- Görev Listesi (Task List): Yapılacak işlerin listesini gösterir. İş paketlerinin altında yer alan eylem gerektiren listedir.

- Zaman Çizelgesi (Timeline): Çizelgenin üstünde bulunan projenin zamanlarını gün, hafta, ay ve yıllık bazda gösteren bir özelliktir.

- Şimdi Çizgisi (Dateline): Genellikle kırmızı renkli dik bir çizgi olarak gösterilir. Bu çizginin solunda kalan işler tamamlanmış olması gerekmektedir.

- Bağlantılar (Dependencies): Bir işe başlamadan önce bir başka işin bitirilmesi gerektiği durumlarda kullanılan terimdir. Gantt şemasında bu ilişkileri gösterebilmek için bağlantıları da kullanmak mümkündür. Bağlantılar ile görevler arasındaki ilişkiyi görebilir, bu görevlerin erken bitmesi/gecikmesi durumunda bir sonraki görevlerin nasıl etkilenebileceğine dair tarih hesaplaması otomatik olarak yapılabilir.

Gantt şemaları aynı zamanda göreve atanan kaynakları ve atamaların yüzde kaçını yansıtabildiği gibi, bir taban çizgisine kıyasla erken ya da geç tamamlanan faaliyetleri belirlemeye yardımcı olabilir. Gantt şemaları tüm projenin izlenmesini sağlayacağı gibi, projenin belirli bir aşamasının ya da kritik bir sürecinin başarısının izlenmesini de sağlar.

Gantt şemasının diğer avantajları aşağıdaki gibidir:

- İletişim: Güncellenmiş bir Gantt şeması, bir bakışta projenin nasıl ilerlediğini gösterebilir, bu da aynı bilgileri paylaşmak için birden fazla toplantıya olan ihtiyacı ortadan kaldırabilir.

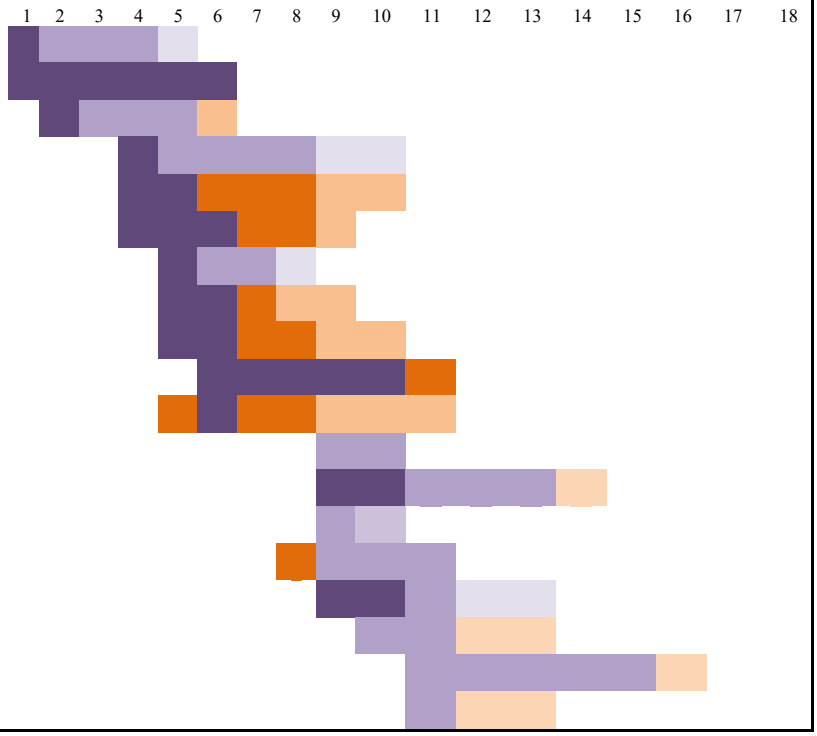
- Motivasyon: Bir projenin büyük resminin ve nerede durduğunu görülmesi proje ekibini tamamlanmaya zorlamak için motive edebilir. Takımın bir sonraki adımı anlamasını sağlar.

- Esneklik: Bir proje başladıktan sonra bir Gantt şemasında değişiklik yapmak kolaydır.

- Verimlilik: Büyük bir projeye giren tüm faaliyetlerin önceden bilinmesi zamanın daha verimli kullanılmasına izin verebilir, diğer ekip üyelerinin yanıtlarını veya geri bildirimlerini beklerken diğer faaliyetler başlatılabilir.

Gantt şemalarının proje yönetiminde kullanmanın en büyük dezavantajı, şemaların çok ayrıntılı ve dolayısıyla hantal olabilmesidir. Bir Gantt şeması, tek bir sayfaya veya ekrana sığan küçük projeler için yararlı ve değerli olsa da, yaklaşık 30'dan fazla aktivite içeren projeler için oldukça hantal hale gelebilmektedir. Gantt şemasını yönetmek kendi başına bir proje haline gelebilir. Bir diğer büyük dezavantaj Gantt şemalarının öncelikleri göstermemesidir. Yeterli zaman varsa, gerçekleşmesinin daha iyi olacağı bir faaliyete karşı yüksek öncelikli bir başka faaliyeti göstermek zordur. Bir Gantt şemasında, bir bütün olarak projeyle ilişkili maliyetleri veya bireysel faaliyetleri belirtmek de zordur. Gantt şemaları bir projenin boyutunu veya iş öğelerinin göreceli boyutunu temsil etmeyeceğinden bir zaman çizelgesinin gerisindeki koşulun büyüklüğü kolayca yanlış iletilebilir. İki proje zamanlamanın aynı süre gerisindeyse, daha büyük projenin kaynak kullanımı üzerinde daha büyük bir etkisi olmasına karşın, Gantt şemaları bu farkı göstermez. Çok sayıda bağımlılığın görüntülenmesi, karmaşık veya okunamayan bir çizelgeye neden olabilir.

Etkinlik	Plan Başlangıcı	Plan Süresi	Fiili Başlangıç	Fiili Süre	Tamamlanma Yüzdesi
Etkinlik 01	1	5	1	4	25
Etkinlik 02	1	6	1	6	100
Etkinlik 03	2	4	2	5	35
Etkinlik 04	4	8	4	6	10
Etkinlik 05	4	2	4	8	85
Etkinlik 06	4	3	4	6	85
Etkinlik 07	5	4	5	3	50
Etkinlik 08	5	2	5	5	60
Etkinlik 09	5	2	5	6	75
Etkinlik 10	6	5	6	7	100
Etkinlik 11	6	1	5	8	60
Etkinlik 12	9	3	9	3	0
Etkinlik 13	9	6	9	7	50
Etkinlik 14	9	3	9	1	0
Etkinlik 15	9	4	8	5	1
Etkinlik 16	10	5	10	3	80
Etkinlik 17	11	2	11	5	0
Etkinlik 18	12	6	12	7	0
Etkinlik 19	12	1	12	5	0
Etkinlik 20	14	5	14	6	0



Gantt Şeması

1.1.2.5. Program Değerlendirme ve Gözden Geçirme Tekniği (PERT)

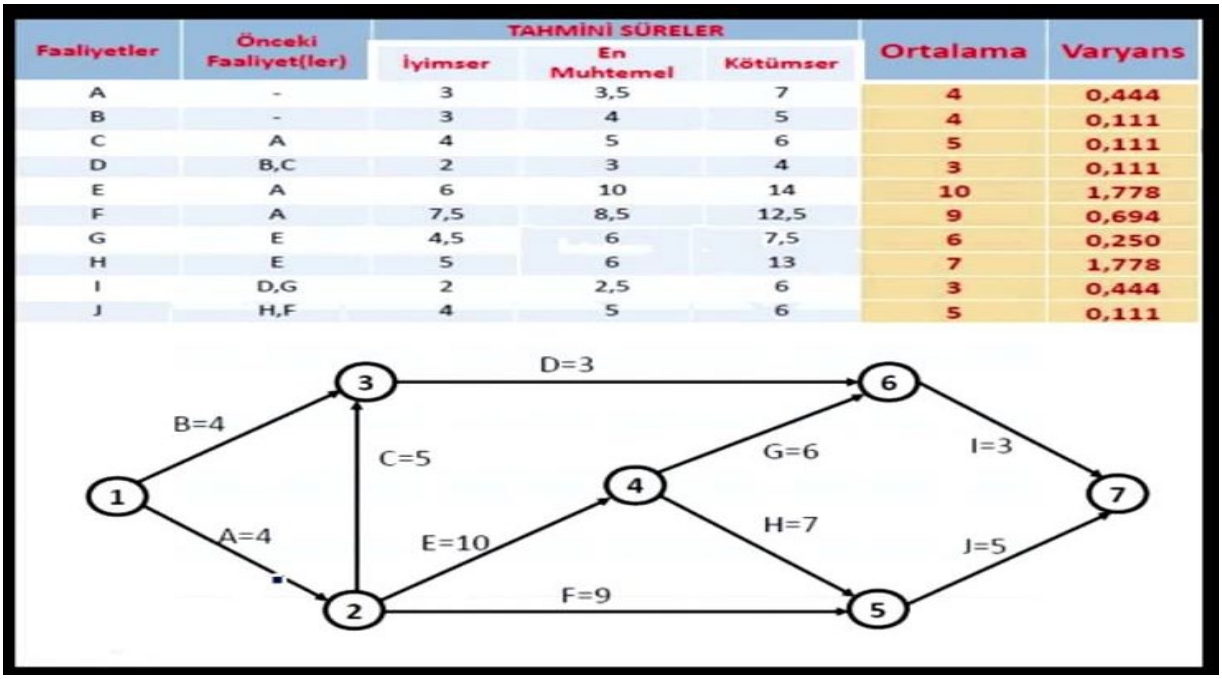
PERT ilk olarak 1950'lerde Polaris denizaltı projesinde Amerikan donanması için kullanılmış ve PERT diyagramlarının kullanılması donanmaya 2 yıllık bir geliştirme süresi kazandırmıştır. PERT, belirsizliğin fazla olduğu araştırma-geliştirme projelerinde kullanılmak üzere geliştirilmiş, fakat daha

sonra farklı sektörlerde de yaygın olarak kullanılmıştır. Dunne ve Klementowski (1982:77), yapılan araştırmalara göre, araştırma-geliştirme projeleri yöneticilerinin, PERT'i en iyi proje planlama tekniği olarak nitelendirdikleri belirtmektedir.

PERT, proje yönetimi ve analizi için kullanılan istatistiksel bir araçtır. Yönetim açısından PERT, planlamanın nasıl yapılması gerektiğini belirtir. Yönetime, koşullar değiştiğinde planlamanın güncel kalmasını sağlayacak gerekli araçlar sunmaktadır. Plandan sapmanın yaratacağı etkileri yönetimin önceden görmesini sağlayarak, olası problemler ortaya çıkmadan önce düzeltici önlemlerin alınmasına imkan tanımaktadır (Thierauf, 1978:173; Stevenson, 1996:784).

PERT diyagramı, grafik temelli olup, projedeki aktiviteleri dokümante ve analiz etme yöntemdir. Projedeki faaliyetler düğümler (nodes) ve faaliyetler arasındaki arası bağlantılar kenarlar (edges) ile gösterilir. Gantt şeması, faaliyetlerin paralelliğini gösterir; ancak faaliyetler arasındaki etkileşimi (hangi faaliyetin hangi faaliyetlere dolaylı olarak bağlı olduğunu) göstermekte yeterli değildir. PERT, projedeki faaliyetlerin yapılış sıralamasını daha net gösterir.

Faaliyetler	Önceki Faaliyet(ler)	TAHMİNİ SÜRELER			Ortalama	Varyans
		İyimser	En Muhtemel	Kötümser		
A	-	3	3,5	7	4	0,444
B	-	3	4	5	4	0,111
C	A	4	5	6	5	0,111
D	B,C	2	3	4	3	0,111
E	A	6	10	14	10	1,778
F	A	7,5	8,5	12,5	9	0,694
G	E	4,5	6	7,5	6	0,250
H	E	5	6	13	7	1,778
I	D,G	2	2,5	6	3	0,444
J	H,F	4	5	6	5	0,111



PERT Şeması

PERT analizinde projenin başlangıcından bitimine giden yollardan en yüksek toplam beklenen (ortalama) süreye sahip yol, kritik yoldur. Projenin beklenen tamamlanma süresi, kritik yolun toplam beklenen süresidir. Ancak gerçek yaşam problemlerinde bir projeyi oluşturan faaliyetlerin kesin sürelerini bilmek mümkün değildir. Faaliyet süreleri bir olasılık dağılıma sahip rassal değişkenlerdir. Eğer projenin faaliyetlerinin tamamlanma süresi kesin olarak bilinmiyorsa, projelerin verilen bir termin süresi içinde tamamlanıp, tamamlanamayacağının olasılık tahmini için PERT kullanılabilmektedir. Diğer bir eleştiri de, projenin başlangıcı ile sonu arasında bulunan faaliyetlerin süre ve varyanslannın art arda toplanarak bulunmasının, bu faaliyetlerin bağımsız olduğu ile ilgilidir. (Öcal, 1991:97; Sauls, 1978:30). Cottrell'in (1999:17), PERT'le ilgili olarak belirttiği diğer bir önemli sorun, her faaliyetin, en iyimser, en kötümser ve en olası süre tahminlerini doğru bir biçimde yapmanın oldukça zor olduğu ile ilgilidir, bu tahminler sübjektif yargıya dayanmaktadır. Karmaşıklığı nedeniyle PERT'in uygulanmasının güç ve maliyetli olması bir diğer olumsuzluktur.

Birtakım eksikliklerine rağmen, PERT, yine de yaygın olarak kullanılmaktadır. Proje yöneticilerine, kritik olmayan faaliyetlerden, projenin zamanında tamamlanmasını etkileyen kritik faaliyetlere, kaynakların nasıl aktarılabileceği konusunda yardımcı olmakta ve projedeki belirsizliklerle baş edebilmek için çoklu süre tahmini yapmayı sağlamaktadır.

PERT ve CPM tekniklerinin proje yönetimini daha basit hale getiren en önemli araçlar olduğu bilinen bir gerçektir. Bu araçların günümüzde çok kullanılır hale gelmesi bilişim sektöründeki hızlı

gelişim ile birlikte; bu teknikleri uygulayan paket programların projelerde aktif olarak kullanılmasının bir sonucudur. Paket bilgisayar programlar 1970'li yıllarda özellikle büyük askeri projelerde kullanılmaya başlanmıştır. Ancak o yıllarda bilgisayarların çizim kapasitelerinin yeterli olmaması ve çizici araçların oldukça pahalı olması bu araçların projelerde kullanımını sınırlı kılmıştır. Fakat günümüzde bilgisayarların hem maliyetlerinin düşmesi, yaygınlaşması hem de kapasitelerinin artması ile birlikte birçok farklı endüstride proje yönetimi ile ilgili paket programlar kullanılır hale gelmiştir.

PERT ve CPM'de bu yaklaşımlara özgü çeşitli terimler kullanılır. Bunlar genelde günlük hayatta kullanılan anlamlarından daha fazlasını ifade eder. Bu nedenle, bu terimlerin bilinmesinde yarar bulunmaktadır. Bu terimler;

- Faaliyet: Projenin gerektirdiği görev veya görev grubudur. Faaliyetler için kaynak ve zaman kullanılmaktadır.

- Olay: Bir veya daha fazla faaliyetin tamamlanması sonucu ulaşılan tanımlanabilir durumdur. Olaylar için zaman veya kaynak kullanmaz. Bir olayın ortaya çıkabilmesi için bunun öncesinde tamamlanması gereken faaliyetler bitirilmelidir.

- Mihenk Noktası: (Kilometre Taşı) Projede dikkate değer gelişmeyi gösteren tanımlanabilir ve önemli olaydır.

- Ağ: Faaliyet ve olayları gösteren, birbirine oklarla bağlı düğüm/kutucukların bulunduğu şekildir. Oklar, faaliyetleri veya teknik olarak faaliyetlerin birbirine bağımlılığını göstermektedir. Ağlar, genellikle solda bir "*başlangıç*" kutucuğu ve sağda "*bitiş*" kutucuğu ile çizilmektedir. Okların yönü bağımlılığı göstermektedir. Okun yönü öncül faaliyetten ardıl faaliyete doğrudur.

- Yol: Ağ içindeki herhangi iki olayı birbirine bağlayan faaliyettir.

- Kritik Yol: Projenin başlangıcından sonuna kadar giden ve herhangi bir gecikme olduğunda tüm projenin gecikmesine yol açacak olan faaliyetler serisidir.

- Kritik Zaman: Kritik yol üzerindeki faaliyetlerin tamamlanması için gerekli olan süredir. Faaliyetlerin hangisinin öncül ve hangisinin ardıl olduğunun bilinmesi çok önemlidir. Öncüllük veya ardıllık ilişkisi faaliyetler arasındaki teknik bağıntıları da tanımlamaktadır.

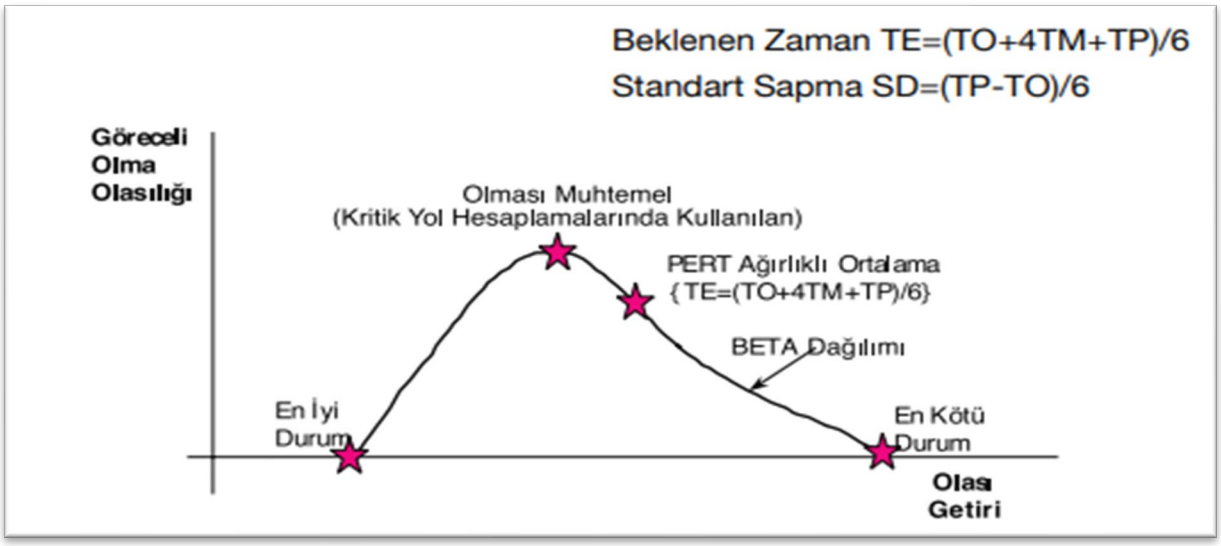
Bu bağıntıların üç temel tipi vardır:

- Zorunlu Bağıntılar: Bu bağıntıların değiştirilmesinin imkânı yoktur. Örneğin boya alınmadan boyama işlemine başlanamaması vb gibi.

- İsteğe Bağlı Bağıntılar: Bu bağıntılar proje yöneticisinin isteğine bağlı olan veya projeden projeye değişebilecek bağıntılardır. Örnek olarak evin boyanması öncesi evden çıkılıp başka bir yere geçilmesi isteğe bağlı bir bağıntıdır. Eğer istenirse boyama esnasında evde kalınabilir.

- Dış Bağıntılar: Bu bağıntılar proje yöneticisinin kontrolünde olmayanlardır. Kritik yol üzerinde boyacıların çalıştığını düşünelim. Proje yöneticisi boyacıların işinin kaç gün süreceğini (eğer bu konuda özel bir uzmanlığı yok ise) bilemez. Bunu belirleyecek olan boyacıların kendileri olacaktır.

Her aktivite süresi için üç farklı tahmin kullanan CPM türü bir tekniktir. Bir görevin tamamlanmasını belirlemek için her bir aktiviteyi tamamlamak için en az üç tahmin gösterilir. Bu tahminler matematiksel formüller kullanılarak tek bir sayıya indirgenir. Ardından klasik bir CPM algoritması uygulanır. PERT ağ yönetimine ilişkin diyagram aşağıda belirtilmiştir. Burada olaylar zaman içindeki noktalar veya faaliyetlerin başlatılması ve tamamlanması için kilometre taşlarıdır. Birincisi en iyi durum (iyimser/TO) senaryodur. Üçüncüsü en kötü durum (kötümser/TP) senaryosudur. İkincisi olması muhtemel (TM) senaryodur. Bu tahmin, büyüklük ve kapsamda benzer projelerden elde edilen deneyimlere dayanmaktadır. Verilen her aktivite için PERT zaman tahminini hesaplamak için aşağıdaki hesaplama uygulanır:



PERT Ağ Yönetimi Şeması

PERT kullanılarak kritik bir yol türetilir. Kritik yol ağdaki en uzun yoldur. Kritik yol, projeyi kısaltabildiği (hızlandırıldığı) veya uzatabildiği (geciktirildiği) yoldur.

Sağlanan örnekte PERT'in CPM'ye göre avantajı formülün üç tahmininin bir beta istatistik dağılımını izlediği ve buna bağlı olarak olasılıkların (ilişki güven seviyeleri ile) toplam proje süresi ile ilişkilendirilebileceği makul varsayımına dayanmaktadır.

Sistem geliştirme projeleri için bir PERT ağı tasarlarken ilk adım, projenin tüm faaliyetlerini ve ilgili olaylarını/kilometre taşlarını ve bunların göreceli sırasını tanımlamaktadır. Örneğin bir olay veya sonuç operasyonel fizibilite çalışmasının sonuçlanması veya kullanıcının ayrıntılı tasarımı kabul ettiği nokta olabilmektedir. Analist herhangi bir faaliyeti göz ardı etmemeye dikkat etmelidir. Buna ek olarak, program kodlama başlamadan önce analiz ve tasarım gibi bazı faaliyetlerin öncesinde diğerleri olmalıdır. Faaliyetlerin listesi PERT ağının detayını belirlemektedir. Analist giderek daha ayrıntılı zaman tahminleri sağlayan birçok diyagram hazırlayabilir.

PERT ile ilgili dikkat edilmesi gereken hususular:

- PERT tahmininin kesinliği kullanılan tahminlerin güvenilirliğine bağlı olarak değişkenlik göstermektedir.

- Yapılacak güncelleme ve düzenleme zaman ve efor gerektirmektedir.

- CPM gibi tüm kaynakların proje için uygun olduğu varsayılarak ilerlenmektedir.

- Kritik yol değişiklikleri tam olarak görülemeyebilir.

1.1.2.6. Kritik Yol Metodu (CPM)

Kritik Yol Metodu (CPM) 1950'li yılların sonlarında DuPont'tan Morgan R. Walker ve Remington Rand'dan James E. Kelley Jr. tarafından geliştirilen bir proje modelleme tekniğidir. Proje Yönetim Enstitüsü (Project Management Institute), bu yöntemi "*Faaliyetler dizisinin analiz edilerek proje süresini tahmin etmek için kullanılan şebeke analiz tekniği, asgari planlama esnekliği miktarına sahiptir. Erken süreler, belirlenmiş özel bir başlama tarihinden ileriye doğru izleye faaliyetlerin takibiyle hesaplanır ve buna ileriye doğru, yine geç başlama süreleri içinde belirlenmiş proje bitim tarihinden geriye doğru faaliyetlerin son bitim süreleri dikkate alınarak başa doğru gelinmesi olan geriye doğru yöntemler kullanılarak hesaplanır.*" olarak tanımlamıştır.

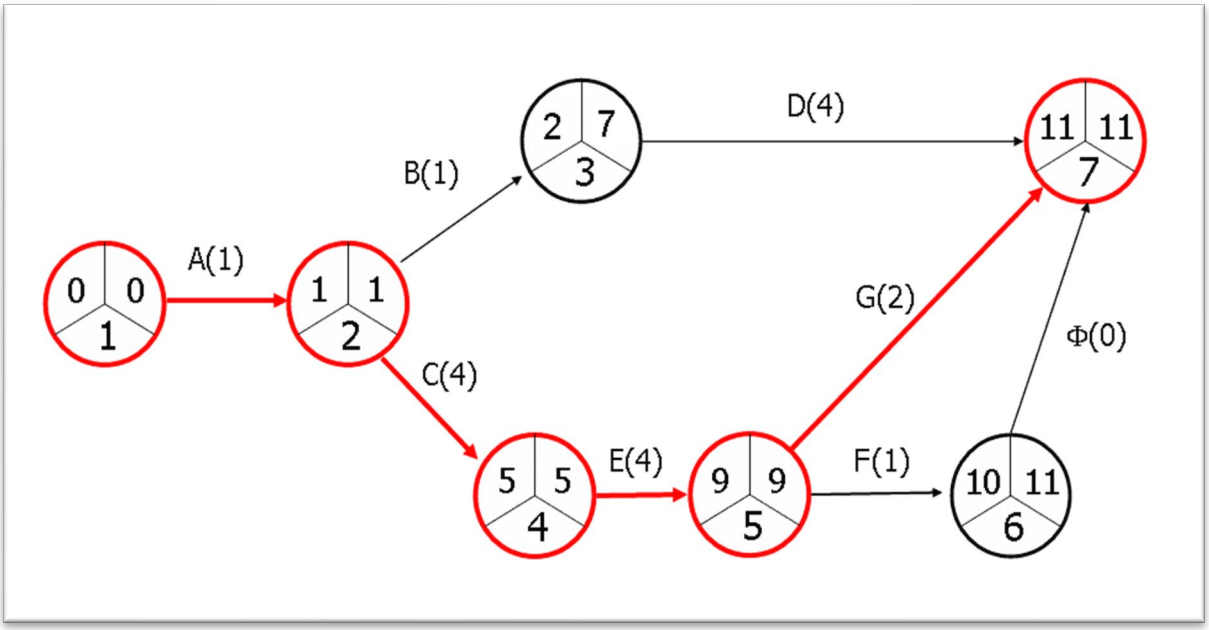
Proje faaliyetlerinin planlanması için kullanılan yöntemlerden biri olan kritik yol metodu, bir projenin tamamlanması için gerekli faaliyetleri tek bir planda toplayarak, faaliyetlerin öncelik ilişkilerini ve sürelerini gösterir ve aynı zamanda projenin kontrolüne yardımcı olur. Kritik yol ise projenin başlangıcı ile bitişi arasındaki en uzun mesafedir. Kritik yol üzerinde bulunan faaliyetlerin sürelerindeki

değişim, proje süresinin değişmesine sebep olur. Basit ve hesaplaması kolay olduğu için dolayı birçok sektörde yapılan projelerde kullanılmaktadır.

Bu yöntem altı safhada uygulanır:

1) Projeyi tanımlanarak faaliyetlerin hiyerarşik yapısını ve sırasını belirlenir.
2) Faaliyetler arasındaki ilişkileri oluşturulur. Hangi faaliyetlerin önce hangilerinin sonra gerçekleştirileceğini belirlenir.
3) Tüm faaliyetler birbirine bağlanarak proje ağını oluşturulur.
4) Her faaliyet için zaman ve/veya maliyet tahminleri yapılır.
5) Ağdaki en uzun yol belirlenir. Bu yol kritik yol olarak adlandırılır.
6) Proje ağı planlama, programlama, izleme ve kontrol faaliyetlerine yardımcı olarak kullanılır.

Bu yöntemde her işlem bir ok ile gösterilir. Her işlem bir düğüm noktası ile başlar ve diğer bir düğüm noktasında biter.



CRM Şeması

Genellikle manipüle edilmemiş proje zaman çizelgelerinde en az bir kritik yol vardır. Projenin mümkün olan en kısa sürede tamamlanması için kritik yollar önemlidir. Kritik yollara dahil edilmeyen faaliyetlerin, sarkma süresi oluşturma riski bulunmaktadır. Sarkma süresi, tüm projenin tamamlanmasını geciktirmeyecek her faaliyetin mümkün olan en son tamamlanma süresi ile öncü tüm faaliyetlere dayanarak mümkün olan en erken tamamlanma süresi arasındaki farktır. Kritik bir yoldaki faaliyetlerin sarkma süresi sıfırdır. Ağ üzerinde ileri- geri çalışılarak, faaliyetler ve proje için mümkün olan en hızlı tamamlanma süresi belirlenir.

Proje yöneticisi dikkatini özellikle kritik yol üzerindeki faaliyetlere vermelidir. Proje yöneticisi projenin başlangıcında hangi faaliyetlerin kritik yol üzerinde olduğunu hangilerinin olmadığını dikkatle kaydetmelidir. Bu durumda iş gücü vb. kaynaklar kritik yol üzerinde olmayan bir faaliyetten kritik yol üzerindeki faaliyetlere aktarılarak kritik süre kısaltılabilir. Bu kararların verilebilmesi için her faaliyetin sarkma süresinin de bilinmesi gereklidir. Projede sarkma süresi olması proje yöneticisinin işini oldukça kolaylaştıran bir durumdur.

PERT yöntemi ile CPR yönteminin farklılıkları aşağıdaki gibidir:

- CPR aktivite odaklı, PERT olay odaklıdır.

- CPR'da süreler bellidir, PERT'te kesin değil olasıdır.

- CPR'da aktiviteler düğüm üzerinde PERT'te ok üzerindedir. PERT'te kilometre taşları düğüm şeklindedir.

- PERT sadece bitiş-başlangıç ilişkilerini (başlangıç-başlangıç, bitiş-bitiş vb.) içerir.

- Düğümler CPR'da dikdörtgen, PERT'te daire şeklindedir.

Kritik yolu kısaltacak önlemler aşağıda belirtilmiştir:

- Kritik etkinlikler üzerine yoğunlaşılması,

- Seri işlerin paralel olarak yeniden planlanması (hızlı izleme),

- Kritik etkinliklerin sürelerinin kısaltılması,

- Fazla mesailerle çalışma sürelerinin uzatılması,

- Erken başlayan işlerin kısaltılması

- En kolay işlerin kısaltılması,

- Çok kaynak kullanan etkinliklerin kısaltılması,

- Maliyeti yüksek işlerin kısaltılması,

- İşletmenin kontrolü altındaki işlerin kısaltılması.

1.1.2.7. Zaman Kutulama Yöntemi

Zaman kutulama yöntemi, Parkinson Yasası denen bir zaman yönetimi anlayışına dayanır. Bu yasaya göre bir iş, yapılması için ayrıldığı süre kadar zaman alır. Eğer sınırlı süre tekniği kullanarak bir görevi yerine getirmek için zaman planlaması yapılırsa o işin o süre içerisinde bitirilebileceğini savunan bir yasadır. Parkinson Yasası'na göre erteleme davranışı yatkınlığı sebebi ile psikolojik olarak bir iş verilen zamana yayılarak yapılır. Dolayısıyla zaman sınırlaması yapılarak planlanan işlerin daha kısa sürede tamamlanması mümkündür.

Zaman kutulama yöntemi, yazılım teslimatlarını nispeten kısa ve sabit bir süre içinde ve önceden belirlenmiş spesifik kaynaklarla tanımlamak ve dağıtmak için bir proje yönetim tekniğidir. Zaman sınırlamasını hayata geçirmek için zaman kutusu çizelgeleri kullanılır. Zaman kutulama tekniğinin temel amacı projenin bilinen bir zaman dilimde sistemin her parçasını zaman kutularına yerleştirerek ele almak ve bu yönüyle zamana, sonuca ya da ürüne odaklı bir sistem geliştirme yaklaşımı sağlamaktadır. Zaman kutusu yönetimi, temel özelliklerin kısa sürede sunulması gereken prototipleme veya hızlı uygulama geliştirme türü yaklaşımları gerçekleştirmek için kullanılabilmektedir. Temel özellikler gelecekteki entegrasyonlar için arayüzler içermektedir. Bu yaklaşımın en büyük avantajı, proje maliyet aşımlarının ve programlı teslimattan kaynaklanan gecikmelerin önlenmesidir. Proje, kalite sürecine olan ihtiyacı ortadan kaldırmaz. Yeni geliştirme araç ve tekniklerinin kullanılması nedeniyle tasarım ve geliştirme aşaması kısaltılmıştır. Test senaryolarının hazırlanması ve test gereklilikleri, son kullanıcı katılımının bir sonucu olarak kolayca yazılmaktadır. Sistem testi ve kullanıcı kabul testi (User Acceptance Test-UAT) normalde birlikte yapılmaktadır.

Zaman kutulama (timeboxing) yöntemi, yazılım geliştirme ve proje yönetimi süreçlerinde kullanılan bir planlama ve zaman yönetimi tekniğidir. Bu yöntemin avantajları ve dezavantajlarına aşağıda yer verilmektedir:

Avantajlar:

Süreklilik ve Disiplin: Zaman kutulama, projenin belirli bir süre zarfında sürekli ve düzenli bir şekilde ilerlemesini sağlar. Bu, ekip üyelerinin işlerini belirli aralıklarla gözden geçirmesini ve iş akışının devam etmesini sağlar.

Hedef Odaklı: Her zaman kutusunda, bir hedef veya öncelik belirlenir. Bu, ekip üyelerinin işlerini bu hedefe odaklanmasını ve dağılmamasını sağlar.

Motivasyon: Belirli bir zaman diliminde bir görevi tamamlama hedefi, ekibi motive edebilir. Zaman sıkıntısıyla çalışmak, işleri tamamlama konusundaki kararlılığı artırabilir.

Önceliklendirme: Önemli görevlere daha fazla zaman ayırmayı ve daha az önemli görevlere daha az zaman ayırmayı sağlar. Bu, kritik işleri tamamlamak için gereken zamana daha fazla odaklanılmasına olanak tanır.

Zaman Yönetimi: Zaman kutulama, zamanı daha iyi yönetme becerisi kazandırır. Görevlerin belirli zaman aralıklarına sığdırılması, zamanın nasıl harcandığını daha net bir şekilde görülmesini sağlar.

İlerlemenin İzlenmesi: Her zaman kutusu sonunda bir görevin tamamlanıp tamamlanmadığı açıktır. Bu, proje ilerlemesini izlemeyi ve gecikmeleri önceden tahmin etmeyi kolaylaştırır.

Ekip İletişimi: Zaman kutulama, ekip içi iletişimi teşvik eder. Her zaman kutusunun başlangıcı veya sonunda, ekip üyeleri ilerlemeyi paylaşabilir ve gerektiğinde işbirliği yapabilirler.

Stres Azaltma: İşleri belirli zaman dilimlerine bölerek, işlerin daha iyi kontrol altında olduğunu hissedilmesine yardımcı olur. Bu da stresi azaltabilir ve daha rahat bir çalışma ortamı yaratabilir.

Dezavantajlar:

Esneklik Eksikliği: Zaman kutulama, görevlerin ve zamanın sıkı bir şekilde planlandığı bir yaklaşımı temsil eder. Bu, beklenmeyen değişikliklere veya acil işlere tepki verme yeteneğini kısıtlayabilir.

Sıkıcı Hale Gelme: Aynı tür görevleri sürekli olarak bu yöntem ile yönetmek, görevleri monoton ve çalışma deneyimini sıkıcı hale getirebilir.

Motivasyon Sorunları: Zaman kutulama, belirli bir zaman diliminde belirli bir görevi tamamlama baskısı yaratabilir ve bu da bazı ekip üyelerinin motivasyon sorunları yaşamasına neden olabilir.

Görevlerin Ayrıntılı Planlanması Gerekliliği: Zaman kutulama yöntemi, görevlerin ayrıntılı bir şekilde planlanmasını gerektirir. Bu, bazı projeler için fazla ayrıntıya ihtiyaç duyulmadığında zaman kaybına neden olabilir.

Risk Yönetimi: Beklenmeyen riskler veya sorunlar ortaya çıktığında, zaman kutulama yöntemi bu tür durumlarla başa çıkmak için sınırlı esneklik sunar.

Kaynak Yönetimi: Zaman kutulama, projelerin ve görevlerin dengeli bir şekilde yönetilmesine yardımcı olur; ancak kaynakları (örneğin, insan gücü ve bütçe) yönetmek için yetersiz olabilir.

Proje Karmaşıklığı: Büyük ve karmaşık projeler için zaman kutulama yöntemi, işleri belirli bir zaman çerçevesine sığdırmak zor olabilir ve bu da işlerin doğru bir şekilde tahmin edilmesini güçleştirebilir.

1.2. Proje İşletimi

Planlama çalışmaları tamamlandıktan sonra, program yöneticisi proje yönetim ofisi ile koordineli yürütülen planlarda, süreçlerde ve prosedürlerde belirtildiği gibi planlanan görevlerin fiili yürütülmesini sağlar. Bu aşamada dikkate alınması gereken bazı önemli noktalar şunlardır:

Koordinasyon: Proje işletimi, proje yönetim ofisi veya proje ofisi ile koordineli bir şekilde yürütülür. Bu, tüm paydaşların ve ekip üyelerinin aynı hedefe doğru ilerlemesini sağlar.

Planlara Uygunluk: Proje işletimi, proje planlarının ve süreçlerinin belirtilen şekilde uygulanmasını içerir. Planlara ve prosedürlere uygunluk, proje başarısı için kritik bir öneme sahiptir.

Görevlerin Yürütülmesi: Proje işletimi aşamasında, proje planında belirtilen görevlerin fiili olarak yürütülmesi sağlanır. Bu, projenin ilerlemesini ve hedeflere doğru ilerleyişi temsil eder.

Süreç ve Prosedürler: Proje işletimi süreçleri, belirli prosedürler ve yönergeleri içerir. Bu süreçler, projenin düzgün bir şekilde yönetilmesini ve kontrol edilmesini sağlar.

Değişiklik Yönetimi: Proje işletimi sırasında, beklenmedik değişiklikler veya sorunlar ortaya çıkabilir. Bu nedenle, değişiklik yönetimi süreçleri de bu aşamada önemlidir.

1.2.1. Proje Açılışı

Proje, sponsor ya da proje yöneticisi tarafından başlatılır. Proje onayının alınması için gerekli bilgiler sponsor ya da proje yöneticisi tarafından toplanmaktadır. Genellikle referans veya projenin amacını, üretilecek sistemdeki paydaşları, proje yöneticisini ve sponsorunu belirten bir proje tüzüğü olarak derlenmektedir.

Proje başlatma belgesinin veya proje talep belgesinin onaylanması, projenin başlaması için yetkilendirmeyi belirtmektedir.

a. Proje Hedeflerinin Belirlenmesi

Proje hedefleri, kaynak kullanımını iyileştirerek stratejik işletme hedeflerine uygun çıktılar oluşturularak hedefe ulaşılmasının sağlanması ve bütçe, maliyet, zaman optimizasyonunu sağlayacak spesifik eylem faaliyetleridir. Tüm proje amaçlarının bu amaca ulaşmak için gereken bir veya daha fazla hedefi olacaktır. Proje hedefi, proje amacına ulaşma yoludur.

Projenin üç kısıtını içeren bir model olan proje üçgeni başarılı bir proje yönetiminin vazgeçilmez unsurudur. Demir üçgen, sihirli üçgen ve proje saç ayağı olarak da bilinen model; zaman, maliyet ve kapsam sabitlerini kalite çemberi içerisinde yönetmektedir.

Kapsam, projenin hangi hedefe ulaşmayı amaçladığı sorusunu cevaplar. Sponsor veya müşteri proje sonucunda hangi ürün veya hizmeti elde etmek amacında olduğu belirlenmektedir. Zaman, projenin tamamlanmasının ne kadar zaman alacağı sorusunu cevaplamaktadır. Diğer bir deyişle ürün veya hizmetin gerçekleştirilmesine ilişkin takvimi ortaya koymaktadır. Başlangıçta kararlaştırılan takvime uyulmadığı takdirde müşteri veya sponsora tazminat ödenmek zorunda kalınabilir. Bu da maliyetlerin artmasına neden olacaktır. Proje yöneticisinin görevi her aşamada takvim ile gerçekleşen işin uyumunun takip edilmesidir. Maliyet, projeyi tamamlamanın maliyetinin ne olacağı sorusunu cevaplamaktadır. Yani eldeki kaynakların çalışır bir sisteme dönüştürülmesinin maliyetini ifade etmektedir. Burada maliyet kelimesi, analist veya programcıların ücretleri, özel ekipmanların satın alınma, kira giderleri, yazılım donanım ve bilgisayar ağları için yapılan harcamalar, yönetim ve kırtasiye giderleri gibi projenin yürütülmesi ile doğrudan ilgili konu başlıklarını kapsamaktadır.

Kapsam, zaman ve maliyet kısıtlarının her üçünü kalite çemberi içerisinde yönetilebilmesi iyi bir proje yönetimi için doğru yolda olunduğunu gösterir. Bu üçgende herhangi bir sabit değiştiği zaman diğer iki sabitinde değişmesi gerekir. Bir projeyi üç ayaklı (kapsam, zaman, maliyet) bir tabureye benzetmek mümkündür. Ayaklardan bir tanesinin olmaması taburenin devrilmesine yol açacaktır. Örneğin; bir projenin kapsamı genişletilirse bu ölçüde zaman ve maliyetin de artması gerekir.

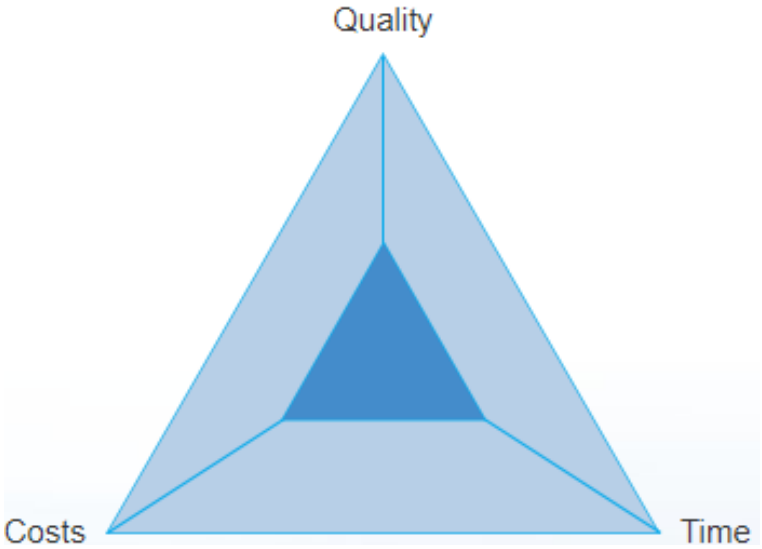
Projelerde öncelikler çoğu durumda birbirine bağımlıdır. Bir önceliği değiştirmek diğerlerini etkilemektedir. Örneğin;

- Proje daha önce bitirilmeli: Etkiler daha fazla maliyet veya daha küçük kapsam/daha az kalite,

- Projenin kapsamı daha büyük olmalı: Projenin daha fazla zamana veya daha fazla maliyete ihtiyacı var,

- Projenin maliyeti daha düşük olmalı: Projenin daha fazla zamana ihtiyacı var veya kapsam daha küçük/daha az kaliteye sahip.

Orta kalite (endüstri standartı), zaman öncelikli ve maliyetin önemsiz olduğu senaryo

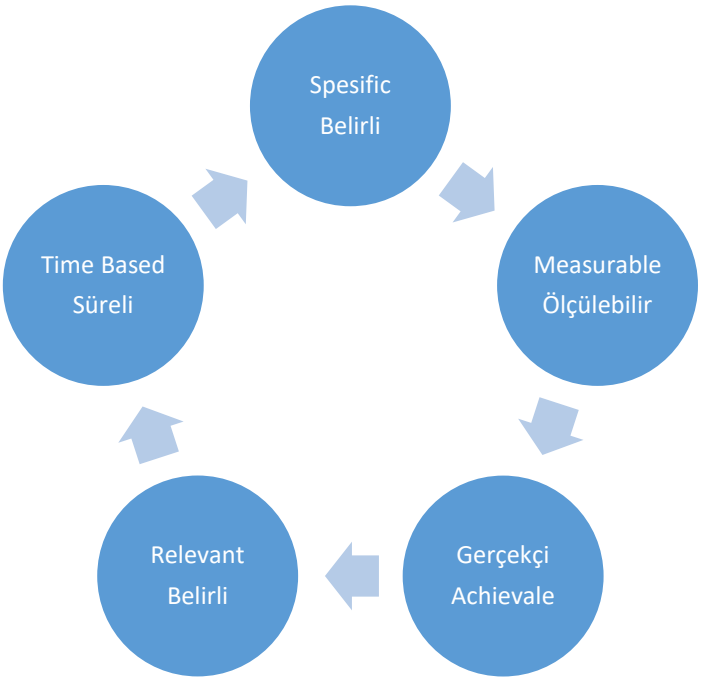


Zaman, kalite ve maliyetin orta seviyede tutulduğu senaryo

Maliyetin öncelikli olduğu senaryo

Zaman + Maliyet + Kapsam = Kalite

Bir projenin belirli (spesific), ölçülebilir (measurable), başarılabilir/gerçekçi (attainable), belirli, ilgili/uygun (relevant) ve süresi (time based) net olarak tanımlanmış sonuçları olmalıdır. Bu beş sonuç SMART metodunu oluşturur. SMART uygulandığında, ölçülebilir ve doğrulanabilir hedefleri göstermektedir.



SMART Şeması

- Belirli (spesific), ulaşılmak istenen hedefin ortaya kesin bir şekilde konulması gereklidir. Genel ifadelerin yerini, özel ifadeler almalıdır. Bunun yanı sıra, neden önemli olduğu ve hangi kaynaklara ihtiyaç duyulduğu belirlenmelidir. Örneğin, bir firmanın kar marjını arttırmayı hedeflemesi genel bir düşüncedir. Hedefi belirli bir hale getirmek için rakamsal ifadelerle nicelik kazandırmak gerekecektir. Örneğin;

Hedef: Yeni bir web sitesi oluşturmak.

SMART Hedefi: Yeni bir e-ticaret web sitesi oluşturmak, ürünlerin çevrimiçi satışını desteklemek ve müşterilere daha iyi bir alışveriş deneyimi sunmak.

- Ölçülebilir (measurable), ulaşılmak istenen hedefe ulaşılıp ulaşılamadığını ya da ulaşılması için ne kadar zaman veya emek harcanacağını her an bilmek için koyulan hedef ölçülebilir olmalıdır. Örneğin, kişinin işini büyütmek gibi genel bir hedefi olmamalıdır. Bir yıl içinde her 3 ay içinde ölçülebilir bir büyüme oranının belirlenmesi, işinin ileri bir boyuta taşınması açısından yararlı olacaktır. Örneğin;

Hedef: Daha fazla müşteri memnuniyeti sağlamak.

SMART Hedefi: Müşteri memnuniyetini artırmak için müşteri anketi sonuçlarını %20 artırmak.

- Başarılabilir/Gerçekçi (achievale), SMART analizi yapılırken hedefte olması gereken diğer bir özelliktir. Mevcut imkanlarla ulaşılabilecek gerçekçi bir hedef olmalıdır. Örneğin; bir işletmenin 5 aylık cirosunu 1 ayda elde etmesi zordur. Çalışan kanaatlerini de göz önünde bulundurarak firmanın başarabileceği hedefler ortaya konmalıdır. Ütopik düşüncelerden kaçınılmalıdır. Örneğin;

Hedef: Şirket içi eğitim programını geliştirmek.

SMART Hedefi: Şirket içi eğitim programını geliştirmek için uzman bir eğitim ekibi kurmak ve kaynakları tahsis etmek.

- İlgili/Uygun, hedefleri belirlerken ana amaca uygun hedefler belirlenmelidir. Örneğin, bir işletme ticaretini Çin ile yapıyorken ve başka bir sahaya girmeyi düşünmüyorken, personellerine İspanyolca öğretmeye çalışması hedefleriyle alakasız olmaktadır. Örneğin;

Hedef: Şirketin pazar payını artırmak.

SMART Hedefi: Şirketin pazar payını artırmak için yeni ürün geliştirme ve pazarlama stratejileri oluşturmak.

- Süreli (time based), hedefler belirlenirken belli bir zaman sınırı koyulmalıdır. Eğer zaman sınır koyulmazsa hedefe ulaşmak için doğru bir planlama yapılamamaktadır. Örneğin, bir işletmenin satış hedefinin 1 yıla yayılarak genel olarak incelenmesi basit bir yönetim şekli olacaktır. Gelir-gider dengesinin sağlanması amacıyla rakamların aylık takip edilmesi gerekmektedir.

Hedef: Bir ürünün lansmanını yapmak.

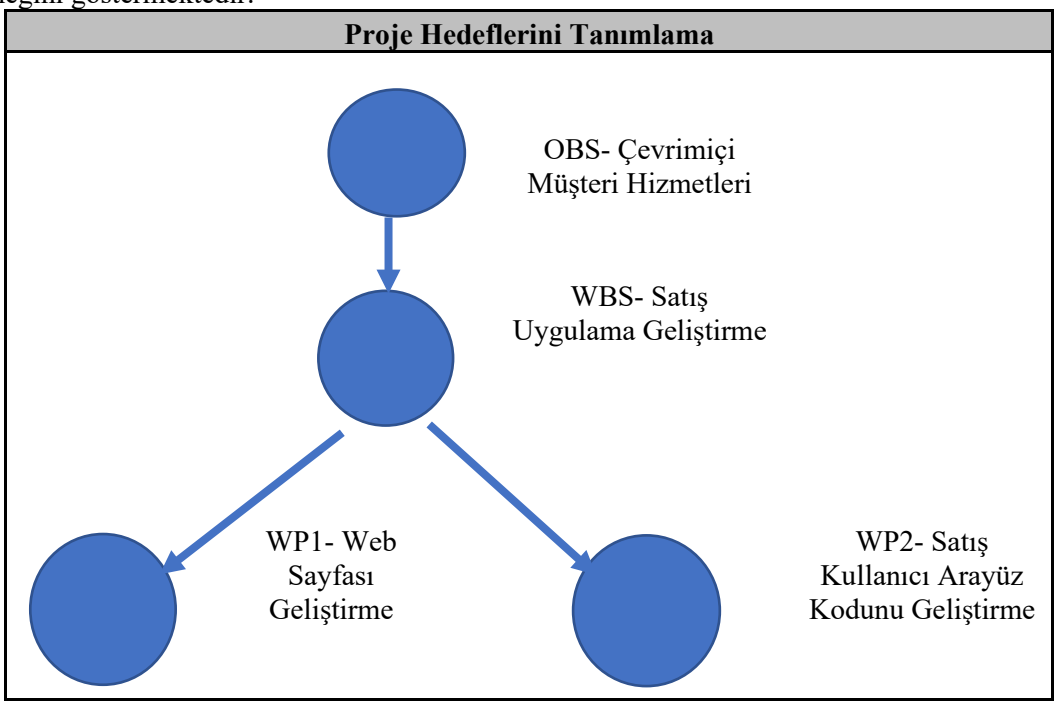
SMART Hedefi: Yeni ürünün lansmanını önümüzdeki 6 ay içinde gerçekleştirmek ve pazarlama kampanyasını Mart sonuna kadar başlatmak.

Yukarıda yer verilen örnekler, SMART kriterlerini her bir adım için nasıl uygulayabileceğinizi göstermektedir. SMART hedefleri belirlerken, hedeflerin spesifik, ölçülebilir, ulaşılabilir, ilgili ve zaman sınırlı olduğundan emin olmak, projenin başarı şansını artırabilir.

Proje hedeflerinin tanımlanması için yaygın kabul gören bir yaklaşım, nesne kırılım yapısı ile başlamaktır. Çözümün bireysel bileşenlerini ve birbirleriyle olan ilişkilerini hiyerarşik bir şekilde grafiksel olarak veya bir tabloda göstermektedir. Bir nesne kırılım yapısı, özellikle organizasyonel gelişim gibi somut olmayan proje sonuçları ile uğraşırken teslim edilebilir bir materyalin göz ardı edilmemesini sağlamaya yardımcı olabilmektedir.

Bir nesne kırılım yapısı derlendikten veya bir çözüm tanımlandıktan sonra, proje sırasında nesne kırılım yapısı öğelerini oluşturması için gerekli olan tüm görevleri yapılandırılması için bir iş kırılım yapısı tasarlanmaktadır. Bir iş kırılım yapısı, projeyi yönetilebilir iş birimleri açısından temsil edilmekte, projede merkezi bir iletişim aracı olarak hizmet etmekte ve maliyet ve kaynak planlamasının taban çizgisini oluşturmaktadır.

Bir nesne kırılım yapısının aksine iş kırılım yapısı oluşturulacak çözümün temel öğelerini içermez. Bunun yerine münferit çalışma paketlerini göstermektedir. Bir iş kırılım yapısı süreç odaklı ve aşamalıdır. İş kırılım yapısının ayrıntı düzeyi, proje sponsoru, proje yöneticisi ve proje ekibi üyeleri arasında ayrıntılı hedeflerin müzakere edilmesi için temel oluşturmaktadır. Aşağıdaki şekil bu sürecin bir örneğini göstermektedir:



Proje Hedefleri Tanımlama Şeması

İş kırılım yapısı, gerekli tüm görevleri listelemekte ve bunları yönetilebilir ve kontrol edilebilir birimler halinde gruplandırmaktadır. Her bir münferit çalışma paketinin ayrı bir sahibi ve ana hedeflerin bir listesi ile ek hedefleri olmalı ve kapsam dışı hedeflerin bir listesi olmalıdır. Münferit çalışma paketlerinin spesifikasyonları diğer çalışma paketlerine bağımlılıkları, performans ve amaç başarısının nasıl değerlendirileceğinin bir tanımını içermelidir. Bir iş kırılım yapısı örneği aşağıdaki şekilde gösterilmektedir.

İş kırılım yapısı ve ilgili çalışma paketleri ile hatırlanması gereken temel konular:

- En üst iş kırılım seviyesi nihai teslimatı veya projeyi temsil etmektedir.

- Alt teslimatlar bir işletmenin bölümüne veya birimine atanan çalışma paketini içermektedir.

- Bir iş kırılımının tüm öğelerinin aynı seviyede tanımlanması gerekmemektedir.

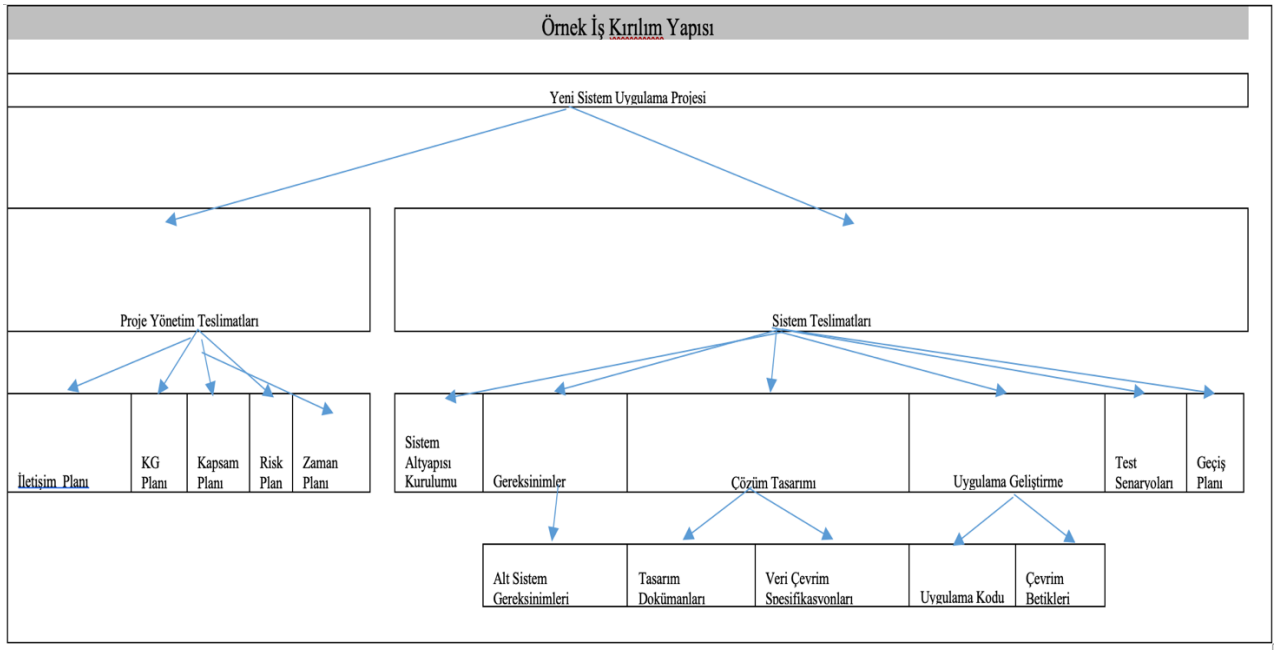
- Münferit çalışma paketleri alt teslimatları üretmek için gereken görevlerin işini, süresini ve maliyetlerini tanımlamaktadır.

- Münferit çalışma paketleri ortalama 10 günü aşmamalıdır.

- Münferit çalışma paketlerinin iş kırılımında birbirinden bağımsız olması gerekmektedir.

- Münferit çalışma paketleri benzersizdir, iş kırılımı genelinde çoğaltılmamalıdır.

İletişimi desteklemek için genelde görev listeleri kullanılmaktadır. Görev listesi münferit çalışma paketleri ile ilgili olarak gerçekleştirilecek eylemlerin listesidir. Atanmış sorumlulukları ve son tarihleri içermektedir. Görev listesi, proje ekibi üyelerine operasyonel planlama ve anlaşmalarda yardımcı olmaktadır. Bu görev listeleri genellikle bir projenin planlama aşamasında bir proje zaman çizelgesinde derlenmektedir. Projenin kontrol aşamasında, çalışma paketlerinin ilerlemesi ve tamamlanmasının takibi için kullanılmaktadır. Proje zaman çizelgeleri yaşayan belgelerdir ve bir çalışma paketinin görevleri, başlangıç ve bitiş tarihleri, tamamlanma yüzdesi, görev bağımlılıkları ve bu görevler üzerinde çalışması planlanan kişilerin kaynak adları belirtilmektedir.



İş Kırılım Şeması

b. İş Olurluk Analizi

İş olurluk incelemesi, bir işletmenin söz konusu projenin devam edip etmeyeceğine karar vermesi için gerekli bilgileri sağlamaktadır. İşletme ve yatırım büyüklüğüne bağlı olarak iş olurluk incelemesinin geliştirilmesi projenin ilk adımı ya da öncüsüdür. İş olurluk incelemesi, projenin

başlatılma ve işletilme gerekçeleri ile faydalarını açıklayacak ayrıntıyı kapsamalıdır. İş olurluk analizi projenin her safhasında önemli bir karar ölçütüdür. Proje yaşam döngüsü içerisinde değişen koşullar karşısında iş olurluk analizi geçerliliğini yitirdiğinde proje sponsoru veya BT yönlendirme komitesi tarafından projenin devam edip etmeyeceği değerlendirilmelidir. Eğer iş olurluk analizinin değişmesi durumunda proje bölüm planlama ve onay süreci işletilmelidir.

Modern projelerde, özellikle agile metodolojilerinin (çevik yazılım geliştirme) kullanıldığı ortamlarda, iş olurluk analizi esnek ve sürekli güncellenebilir bir süreç halini alır. Bu analizler, yalnızca projenin teknik açıdan uygulanabilirliğini değil, aynı zamanda projenin stratejik hedeflerle uyumunu da inceler. Veri analitiği (data analytics) ve yapay zeka (AI) gibi modern teknolojiler, iş olurluk analizi süreçlerinde projelerin başarı şanslarını daha doğru tahmin etmeyi sağlar. Bu sayede, projenin tüm işlevsellikleri ve riskleri daha doğru bir şekilde değerlendirilebilir ve işin yapılabilirliği daha iyi anlaşılabilir.

Bilgi sistemleri denetçisi, işletmenin fizibilite çalışmaları sırasında kullanılan iş olurluk incelemelerini nasıl tanımladığını ve bilgi sistemleri geliştirmesiyle ilgili projeler için yatırım getirisi ile ilgili sonuç tespitlerini anlamalıdır. İşletme yatırım getirisi hedeflerini tutarlı bir şekilde karşılayamazsa, bu durum sistem geliştirme yaklaşımında ve ilgili proje yönetimi pratiklerinde zayıflıklara işaret edebilir.

Bilgi sistemleri denetçisinin rolü, iş olurluk incelemesi ve yatırım getirisi analizi bağlamında daha kapsamlı bir şekilde aşağıdaki maddeler halinde ele alınmaktadır:

İş Olurluk İncelemesi ve Analizi:

Bilgi sistemleri denetçileri, işletmenin yeni projeleri başlatmadan önce iş olurluk incelemesi ve analizi süreçlerini destekler ve değerlendirir. Bu incelemeler, projenin işletilme ve sürdürülme gerekliliklerini ve faydalarını ortaya koymalıdır.

Risk Değerlendirmesi:

Bilgi sistemleri denetçileri, iş olurluk incelemesi ve yatırım getirisi analizi sırasında olası riskleri değerlendirirler. Bu riskler, projenin maliyetlerini, zamanlamasını ve başarı olasılığını etkileyebilir.

Proje risklerinin yönetilmesi aşamasında, yapay zeka tabanlı risk analiz araçları (AI-based risk analysis tools) ve big data (büyük veri) çözümleri kullanılarak, projenin tüm risk faktörleri tahmin edilebilir ve daha doğru bir risk değerlendirmesi yapılabilir. Günümüzde cloud computing (bulut bilişim) ve blockchain gibi teknolojiler, projelerin güvenlik, operasyonel hatalar veya zaman aşımı gibi risklerini daha erken aşamada tespit etmeye yardımcı olabilir. Ayrıca, projelerin devam etme kararı alınmadan önce bu risklerin etki derecesi ve projeye olan potansiyel zararları hesaplanarak, stratejik bir karar alınması sağlanır. Risk değerlendirmesi sürecinde bilgi sistemleri denetçilerinin bu kavramlara dikkat etmesi önem arz etmektedir.

Yatırım Getirisi Hesaplamaları:

Bilgi sistemleri denetçileri, projenin işletme açısından maliyetlerini ve getirilerini hesaplarlar. Yatırım getirisi analizi, projenin maliyet etkinliğini ve işletme açısından karlılığını değerlendirmeye yardımcı olur.

Modern projelerde, ROI hesaplaması yalnızca finansal verilerle sınırlı kalmaz; projelerin işletme değeri (business value) ve müşteri memnuniyeti gibi unsurlar da dikkate alınır. Veri madenciliği (data mining) ve machine learning (makine öğrenmesi) gibi araçlar, geçmiş projelerin verilerini analiz ederek gelecekteki yatırım getirisi hesaplamalarının doğruluğunu artırabilir. Bu araçlar, müşteri talepleri, pazar trendleri ve sistem performansı gibi faktörleri değerlendirerek daha kapsamlı bir ROI hesaplaması yapmayı mümkün kılar. Yatırım getirisi hesaplamaları sürecinde bilgi sistemleri denetçilerinin bu kavramlara dikkat etmesi önem arz etmektedir.

Raporlama ve İzleme:

Bilgi sistemleri denetçileri, iş olurluk incelemesi ve yatırım getirisi analizini raporlarlar ve bu süreçleri izlerler. İşletme yönetimine projelerin potansiyel faydaları ve riskleri hakkında düzenli olarak bilgi sunarlar.

Raporlama ve izleme süreçleri, iş olurluk incelemesi ve yatırım getirisi analizinin başarısını değerlendirmek için kritik araçlardır. BI (Business Intelligence) araçları ve dashboard (gösterge panelleri) çözümleri, projelerin ilerleyişini ve performansını daha şeffaf bir şekilde takip etmek için kullanılmaktadır. Bu araçlar, projelerin zaman ve bütçe takibini yaparken, aynı zamanda risklerin, kaynak kullanımının ve diğer kritik faktörlerin izlenmesini de sağlar. Modern raporlama teknikleri, gerçek zamanlı veri (real-time data) kullanarak projelerin güncel durumunu yöneticilere sunar ve hızlı karar alınmasını kolaylaştırır. Raporlama ve izleme sürecinde bilgi sistemleri denetçilerinin bu kavramlara dikkat etmesi önem arz etmektedir.

İşletme Hedefleri ile Uyumluluk:

Bilgi sistemleri denetçileri, projelerin işletme hedefleriyle uyumlu olduğundan emin olurlar. Projeler, işletmenin stratejik hedeflerini desteklemeli ve gereksinimlerini karşılamalıdır.

Bir yazılım projesi, yalnızca teknik gereksinimleri karşılamakla kalmamalı, aynı zamanda işletme hedefleriyle de uyumlu olmalıdır. Modern yazılım projelerinde, işletme hedefleri ve yazılım gereksinimleri arasındaki uyumu sağlamak için agile (çevik) ve scrum metodolojileri gibi esnek yönetim yaklaşımları kullanılır. Bu metodolojiler, işletmenin stratejik hedeflerine yönelik esnek ve hızlı çözümler sunmayı amaçlar. Ayrıca, cloud-based (bulut tabanlı) yazılım çözümleri, işletme hedefleriyle uyumluluğu artırarak, projelerin sürekli olarak optimize edilmesine ve güncellenmesine olanak tanır.

Fizibilite Çalışmaları:

Bilgi sistemleri denetçileri, projelerin fizibilite çalışmalarını değerlendirirler. Bu çalışmalar, projenin teknik olarak, finansal olarak ve operasyonel olarak uygulanabilirliğini değerlendirir.

Fizibilite çalışmaları, yazılım projelerinin başarıyla tamamlanıp tamamlanamayacağını belirlemek için yapılan araştırmalardır. Günümüzde fizibilite çalışmaları sadece teknik açıdan değil, aynı zamanda finansal ve operasyonel açıdan da yapılmaktadır. AI tabanlı tahmin araçları ve big data analizleri, projelerin teknik, finansal ve operasyonel açıdan uygun olup olmadığını belirlemede daha doğru sonuçlar sağlar. Ayrıca, fizibilite çalışmaları, proje risklerini de içeren bir analiz yaparak, projenin başarısı için gereken kaynakların doğru bir şekilde belirlenmesini sağlar. Fizibilite çalışmaları kapsamında bilgi sistemleri denetçisinin bu hususlara da dikkat etmesi verimliliğin artışı bakımından önem arz etmektedir.

Değişiklik Yönetimi ve Revizyonlar:

İş olurluk incelemesi ve yatırım getirisi analizi sonuçlarına dayalı olarak, bilgi sistemleri denetçileri gerektiğinde projelerin değiştirilmesi veya revize edilmesi gerekip gerekmediği konusunda önerilerde bulunurlar.

İş olurluk incelemesi ve yatırım getirisi analizi sırasında yapılan değişiklikler, projelerin yönlendirilmesinde önemli bir rol oynar. Agile ve Scrum metodolojileri, projelerde yapılacak değişikliklere hızlı adaptasyon sağlayarak, yazılım geliştirme sürecinin daha esnek ve sürdürülebilir olmasını sağlar. Continuous integration (sürekli entegrasyon) ve DevOps yaklaşımları, yazılım geliştirme sürecinde yapılan her değişikliğin hızlıca test edilmesini ve entegrasyonunu sağlar. Bu yöntemler, yazılım projelerinin hızla gelişmesine ve değişen iş ihtiyaçlarına uyum sağlamasına olanak tanır.

Etik İlkeler ve Uyum:

Bilgi sistemleri denetçileri, projelerin etik ilkelerle uyumlu olduğundan ve tüm yasal ve düzenleyici gerekliliklere uygun olduğundan emin olurlar.

Proje yönetimi sürecinde etik ilkeler, yazılım geliştirme sürecinin düzgün ve güvenli bir şekilde ilerlemesini sağlar. GDPR (Genel Veri Koruma Yönetmeliği) gibi veri güvenliği yönetmeliklerine uyum, yazılım projelerinin yasal gereksinimlere uygun olarak geliştirilmesini sağlar. Ayrıca, blockchain gibi teknolojiler, yazılım projelerinde veri güvenliğini artırırken, şeffaflık ve uyumluluk açısından büyük avantajlar sunar. Bu tür düzenlemelere ve etik ilkelere uyum, yazılım projelerinin yasal ve toplumsal açıdan kabul edilebilirliğini sağlar. Bilgi sistemleri denetçisinin bu hususlara da hakim olması gerekmektedir.

İletişim ve Paydaş Yönetimi:

Bilgi sistemleri denetçileri, projelerin paydaşlarla etkili iletişimini destekler ve paydaşların beklentilerini anlamalarına yardımcı olur.

Proje yönetimi sürecinde etkili iletişim ve paydaş yönetimi, projenin başarısı için kritik öneme sahiptir. Yapay zeka ve chatbotlar, proje ekibiyle paydaşlar arasındaki iletişimi kolaylaştırırken, hızlı geri bildirim sağlayarak projelerin gelişimini hızlandırır. Ayrıca, paydaş katılımı (stakeholder engagement)) araçları, paydaşların proje süreçlerine dahil edilmesi ve onların geri bildirimlerinin hızlıca alınmasına olanak tanır. Bu araçlar, paydaşların beklentilerine göre projelerin şekillendirilmesini sağlar.

İyileştirme Önerileri:

Bilgi sistemleri denetçileri, iş olurluk incelemesi ve yatırım getirisi analizi sonuçlarına dayalı olarak iyileştirme önerileri sunarlar ve projelerin daha etkili ve verimli bir şekilde yönetilmesini teşvik ederler.

Proje süreçlerinin sürekli olarak iyileştirilmesi, başarıyı artıran en önemli faktördür. Veriye dayalı karar verme (Data-driven decision making) ve AI analitik araçları, proje verilerinden öğrenerek süreçleri optimize etmek için kullanılır. Bu araçlar, proje yöneticilerinin kaynakları daha verimli kullanmalarını ve proje süreçlerinde en iyi uygulamaları belirlemelerini sağlar. Proje performansının analiz edilerek sürekli iyileştirilmesi, yazılım geliştirme süreçlerinin daha verimli hale gelmesini sağlar. Bilgi sistemleri denetçisinin bu hususlarda da bilgi sahibi olması beklenmektedir.

c. Projenin Yürütülmesi

Projenin yürütülmesi kapsamında dokümantasyon, toplantılar, gözetim faaliyetleri ele alınmaktadır. Proje büyüklüğüne, karmaşıklığına ve etkilenen taraflara göre proje yöneticisi/sponsoru tarafından; proje ekibi ve proje yöneticisi arasında gerçekleştirilen birebir toplantılar, proje ekibinin iş birliği ve katılımıyla gerçekleştirilen çalıştaylar, proje yöneticisinin proje ekibine aktarımlarını ve proje iletişiminin yürütülmesini sağlayan başlangıç toplantıları yollarından en az birini seçerek başlatılabilir.

Projenin yürütülmesi aşamasında ele alınan bazı ana faaliyetler aşağıdadır:

Dokümantasyon:

Proje dokümantasyonu, projenin ilerlemesini ve sonuçlarını kaydetmek için kullanılır. Bu dokümanlar, proje planlarını, ilerleme raporlarını, risk yönetimi belgelerini ve diğer önemli bilgileri içerebilir. Dokümantasyon, proje yöneticisi ve paydaşlar arasında netlik sağlar.

Toplantılar:

Proje ekibi arasında düzenli toplantılar, ilerlemeyi değerlendirmek, sorunları çözmek ve işbirliğini teşvik etmek için önemlidir. Proje yöneticisi liderliğinde yapılan toplantılar, proje ekibini güncel tutar ve kararların alınmasına yardımcı olur.

Gözetim Faaliyetleri:

Projenin ilerlemesini ve performansını izlemek amacıyla gözetim faaliyetleri gerçekleştirilir. Bu faaliyetler, proje hedeflerine uygun olarak ilerlenip ilerlenmediğini değerlendirmek ve gerektiğinde düzeltici önlemleri almak için kullanılır.

Çalıştaylar:

Projeye özgü karmaşık sorunları çözmek veya yaratıcı çözüm önerileri üretmek amacıyla proje ekibi ile birlikte yapılan çalıştaylar düzenlenir. Bu çalıştaylar, farklı uzmanların görüşlerini paylaşmasını ve ekip içi işbirliğini destekler.

Başlangıç Toplantıları:

Proje yürütülmesi aşamasına başlarken, proje yöneticisi/sponsoru tarafından başlangıç toplantıları düzenlenebilir. Bu toplantılarda proje amaçları, hedefler, roller, sorumluluklar ve iletişim planı gibi konular ele alınır. Başlangıç toplantıları, proje ekibi arasında bir anlayış ve motivasyon oluşturmayı amaçlar.

İletişim Yönetimi:

Proje iletişimi, projenin tüm paydaşları arasında etkili bir şekilde yönetilmesini gerektirir. İletişim planı oluşturulur ve güncellenir, paydaşlarla düzenli iletişim kurulur ve bilgi akışı sağlanır.

1.2.2. Projenin Kontrolü ve İzlenmesi

Proje çalışmalarının izlenmesi, proje yönetimi planında tanımlanan performans hedeflerine ulaşılması için projenin ilerlemesinin izlenmesi, gözden geçirme ve düzenleme sürecidir. Kontrol ise düzeltici ya da önleyici eylemleri belirlemeyi ve gerçekleştirilen eylemlerin performans sorununun çözülüp çözülmediğinin belirlenmesi amacıyla eylem planlarının izlenmesini ya da yeniden yapılanmasını içermektedir.

Planlama aşamasında yükleniciler ve tedarikçilere ilişkin ölçütler izlenmektedir. Projenin BT sistemi gereksinimleri, mimarisi, tasarımı, geliştirilmesi, test edilmesi, uygulanması ve üretim operasyonlarına geçişi konusunda entegre ekibin gözetimi önemli bir başarı faktörüdür.

1.2.2.1. Projelerde Kontrol ve İzleme Kavramı

Bir projenin kontrol ve izleme faaliyetleri kapsam, kaynak, kullanım ve risk yönetimini içermektedir.

Proje kontrolü, projedeki faaliyetlerin durumun değerlendirilmesi, proje durumunun planlanan durumla karşılaştırılması ve eğer gerekiyorsa düzeltici önlemlerin alınması için yapılan faaliyetlerdir. Proje kontrolü sayesinde projenin yürütülmesi sırasında sorun yaratabilecek, kritik veya yan kritik faaliyetler üzerinde yoğunlaşmak mümkündür (Monks, 1996:354).

Gerçekleşen ilerlemeyle ilgili sürekli bilgi akışının olması proje yöneticisinin belirsizliklerle baş edebilmesini sağlayacak bir geri-besleme mekanizmasıdır. Gerçekleşen ilerlemeyi planla karşılaştırmakla sapmalar belirlenmekte, problemler önceden tespit edilmekte ve düzeltici hareketler yapılabilmektedir. Çizelge ve teknik alanlarda beklenenden az bir başarı ve maliyetlerdeki sapmaların erkenden tespit edilmesi, yöneticilere önemli konular üzerinde odaklanma imkânı vermektedir. Projenin kontrolü ve güncellenmesi neticesinde alınan sağlıklı kararlar ile projenin başarısındaki sapmalar minimize edilebilmektedir.

Proje kontrolü, etkili yönetimi ve kararı destekleyen formatlarda bilgi iletilerek bir proje veya programın zaman ve maliyet sonuçlarını tahmin edilmesi ve yapıcı bir şekilde etkilemesi için kullanılan veri toplama yönetimi ve analiz süreçlerinden oluşmaktadır.

İyi bir proje yöneticisi, proje kontrolünün ne olduğunu tam olarak kavrayabilmek için üç unsur üzerinde durmalıdır. Bu unsurlar önleme, tespit etme ve eylem olarak sıralanabilir.

- Önleme: Önleme, projenin istenildiği biçimde ilerleyebilmesini sağlamak için proje planından olası sapmaların ortaya çıkmasının engellenmesidir. Bunun gerçekleştirilebilmesi için proje yöneticisinin tüm bilgi birikimini kullanması ve gerektiğinde diğer paydaşlar ile bilgi paylaşımına gitmesi gerekir. Ek olarak iyi bir proje yöneticisinin planlama aşamasında iyi bir yatırım yapması, etkin bir iletişim gücüne sahip olması, risk faktörlerini sürekli olarak izliyor olması, sorunların çözümünde mümkün olduğunca saldırgan bir tavır takınmaması ve yapılacak işleri mümkün olduğunca açık bir dille anlatması gereklidir.

- Tespit Etme: Tespit etme, proje kontrolünün bu boyutunu bir erken uyarı sistemi olarak görmek mümkündür. Proje kontrol sistemi olası sapmalar için bir erken tespit sistemi olmalıdır. Herhangi bir plan sapmasına ne kadar çabuk müdahale edilirse problemin ortadan kaldırılması ve tekrar temel plana dönülmesi de o denli kolay olacaktır. Erken tespit için en önemli faktörler iyi bir izleme sistemi kurulması ve proje sonuçlarının zamanında ölçülmesine olanak verecek iş parçacıklarının en iyi biçimde geliştirilmesidir. Kullanılabilecek tespit sistemlerine performans raporları ve değerlendirme toplantıları örnek olarak gösterilebilmektedir. Burada üzerinde durulması gereken bir diğer nokta da bir sapmayı tespit edebilmek için ilgili faktörün önceden belirlenmiş temel bir değerinin olması gerektiğidir. Sapmalar herhangi bir başarı faktöründe ortaya çıkabilir, paydaşların beklentileri ve kalite söz konusu bu başarı faktörlerinden bazılarıdır.

- Eylem: Eylem, genelde önleme işleminin gerçekleştirilebilmesi için de birçok eylemin hayata geçirilmesi gerekmektedir. Ancak eylemi tespit aşaması ile omuz omuza ilerleyen bir unsur olarak görmek gerekir.

Proje kontrolünün mümkün olduğunca etkin olabilmesi için herhangi bir sapmanın tespitinin çözüme yönelik uygun ve zamanında bir tepkiyi doğurması gerekir. Bu noktada proje yöneticisinin uygulayabileceği üç farklı eylem tipi öne çıkmaktadır. Bu eylem tipleri; düzeltici eylemler, değişim kontrol prosedürleri ve kazanılmış dersler olarak sıralanabilir. Bir proje zaman, maliyet ve kalite hedeflerinden sapmaya başladığında ve proje yöneticisinin çeşitli eylemleri uygulaması kaçınılmaz hale geldiğinde "*Yöneticinin eylem seçenekleri nelerdir?*" sorusu cevaplanmalıdır. Temel bazı değişiklik ve eylem seçenekleri aşağıdaki gibi sıralanabilir:

- Projenin yeniden tahmin edilmesi: Daha önceden tanımlanan tüm tahmin değerleri ikinci kez kontrol edilir. Projenin paydaşlarının maliyet ve zaman kısıtları konusundaki isteklerini karşılamak ve projenin temel amacına ulaşmak için kullanılabilecek alternatif yollar araştırılarak projeye dâhil edilebilir.

- Projeye yeni bireyler eklenmesi: İlk bakışta projeye yeni bireyler eklenmesi hızlı ve kolay bir çözüm gibi görünmekle birlikte, yeni bireyler kendi fikir ve görüşlerini projeye aktarmak istediğinde fikir çatışmaları ortaya çıkabilecektir. İşin gerçekten daha hızlı bir biçimde tamamlanmasına destek olacak bireylerin projeye eklenmesi daha faydalı bir yaklaşımdır.

- Görevlerin kapsamının daraltılması: Bazı durumlarda projenin başarı ile tamamlanması için plan içinde yer alan bazı iş parçacıkları kaldırılabilir. Böylece tamamlanması nerede ise imkânsız düzeye ulaşan iş parçacıkları yerine daha altından kalkılabilir iş parçacıklarına ulaşılması sağlanır. Ancak hangi iş parçacıklarından vazgeçileceği belirlenirken proje hedeflerinden bir sapmanın ortaya çıkmaması gerektiği de göz önünde bulundurulmalıdır.

- Üretkenliğin verimli personel ile arttırılması: Kimi bireyler diğerlerine göre daha üretken ve verimlidir. Yakın çevrede proje ekibinde yer almayan daha üretken elemanlar bulunduğunda, bu elemanların kısa süreli olarak projede yer almasının sağlanması yoluna gidilmelidir. Ayrıca proje ekibi içinde yer alan üretken üyeler de kısa süreli olarak projenin başka aşamalarında çalıştırılabilir.

- Dış kaynak kullanımı: Yoğun çalışmalar sonucu hazırlanan özgün plana göre yapılması gereken bir iş parçacığını veya proje fazını proje ekibinden daha hızlı, daha verimli ve kalite standartlarına daha uygun yapabileceğine inanılan bir dış kaynak bulunduğunda, o iş veya fazın dış kaynağa verilmesinde bir sakınca yoktur.

- Fazla mesai kullanımı: Proje yöneticisi ölçülü ve ihtiyatlı davranmayı ihmal etmeden fazla mesai seçeneğine başvurabilir. Ancak fazla mesai uygulaması proje ekibinin beklendiğinden daha önce yorulmasına sebep olacağından proje ekibinin verimliliğinde bir düşmeye yol açabilir. Ölçüyü aşmaksızın gerektiğinde kullanıldığında projeye katkı yapacak bir eylem biçimidir.

- Bazı işlerin müşteriye aktarılması: Proje bir müşteri için gerçekleştirildiğinde ve proje yüksek maliyetli ama insan kaynakları bakımından kısıtlı olduğunda, bazı işler müşterinin de kabul etmesi hâlinde müşterinin kendisine bırakılabilir.

- Proje amaçlarının yeniden düzenlenmesi: En tehlikeli eylem türüdür. Projenin beklenen sonuçları sağlaması için kapsam daraltması ilk bakışta iyi bir yaklaşım gibi gözükmekle birlikte projenin kalitesinden ödün verme yönünde atılacak bir adım çok kötü sonuçlar doğurabilir. Kalite standartlarını yakalamadan tamamlanan bir proje, projeyi yürüten ekip veya işletme için kötü bir reklam haline gelebilir.

- Proje ekibinin uyumlu çalışması: Tüm işletmelerin süreçlerinde olduğu gibi izleme ve kontrol süreçlerinde de en önemli unsur proje ekibinin uyum içinde çalışmasıdır. Proje yöneticisinin, gerekli durumlarda proje ekip elemanlarına, kendilerinin arkasında olduğu güvenini vermesi elemanların işe daha iyi sarılmalarını sağlayabilir.

Proje kontrolü, proje yönetiminin birçok unsuruna uygulanabilir. Ancak, aşağıdaki unsurlar proje kontrolünün ana alanlarını temsil etmektedir:

- Planlama

- Risk yönetimi

- Maliyet yönetimi

Proje Kontrolünün Avantajları: Proje kontrolünün aşağıdaki faydaları, onu proje yönetiminin vazgeçilmez bir parçası haline getirir:

- Bir projenin toplam maliyetleri, örneğin temel performans göstergelerine (KPI) dayalı etkili karar verme yoluyla düşük tutulabilir.

- Maliyetler ve son tarihler için daha fazla öngörülebilirlik sağlar.

- Bir projenin tüm aşamalarında finansal durumu hakkında daha fazla ve daha iyi içgörüler sağlar.

- Sonuçların ertelenmesini daha kolay önlemeyi sağlar.

- Daha yüksek marjlar sağlar.

- Proje yönetimi ve sonuç garantileri söz konusu olduğunda daha iyi bir itibar sağlar.

- Artan bir çalışan memnuniyeti duygusu sağlar.

- Etkili proje yönetimi, daha az olgun proje yönetimi yeteneklerine sahip proje ekiplerine göre rekabet avantajı sağlar.

Bilgi sistemleri denetçisi, aşağıdaki proje yönetimi faaliyetlerinin yeterliliğini gözden geçirmelidir:

- Proje komiteleri/kurulu gözetim seviyesi,

- Risk yönetim yöntemleri,

- Sorun yönetimi,

- Maliyet yönetimi,

- Planlama ve bağımlılık yönetimi süreçleri,

- Raporlama süreçleri,

- Değişim kontrol süreçleri,

- Paydaş yönetiminin katılımı,

- Onay süreci.

1.2.2.2. Kapsam Değişikliklerinin Yönetimi

Projelerin kapsamını yönetmek, bir iş kırılım yapısı şeklinde dikkatli bir belgeleme gerektirir. Bu belgeler proje planının veya proje taban çizgisinin bir parçasını oluşturur. Kapsamdaki değişiklikler nerdeyse sürekli gerekli faaliyetlerde değişiklikleri ve son teslim tarihlerini ve bütçeyi etkiler. Bu nedenle, resmi değişiklik talepleri de dahil olmak üzere değişiklik yönetim sürecinin bir parçası olmalı ve dökümante edilerek saklanmalıdır. Değişiklikleri sadece proje paydaşları önerebilmektedir. Proje yöneticisi her değişiklik talebini kapsam, bütçe ve program açısından değerlendirmektedir. Ardından değişiklik danışma kurulu da değişiklik talebini değerlendirmektedir. Değişiklik kabul edilirse proje planı güncellenir. Güncellenen proje planı proje sponsoru tarafından resmi olarak onaylanmalıdır.

Kapsam değişikliklerinin yönetimi, projenin başlangıcından sonuna kadar kapsamının doğru bir şekilde tanımlanması ve herhangi bir değişiklik veya eklemenin düzenli ve kontrollü bir şekilde yönetilmesini içerir. Bu sürecin ana adımları aşağıda detaylı olarak maddeler halinde ele alınmıştır:

İş Kırılım Yapısının Oluşturulması: Proje başlangıcında, iş kırılım yapısı (Work Breakdown Structure - WBS) oluşturulur. Bu, projenin tüm görevlerini ve alt görevlerini ayrıntılı bir şekilde listeleyen bir belgedir. WBS, proje kapsamının anlaşılmasına yardımcı olur.

Kapsam Değişikliklerinin Belgeleme: Herhangi bir kapsam değişikliği veya ek, resmi bir değişiklik talebi formu kullanılarak belgelenir. Bu talepler genellikle proje paydaşları tarafından sunulur ve değişikliklerin nedeni ve etkileri detaylı bir şekilde açıklanır.

Değişiklik Değerlendirmesi: Proje yöneticisi, her değişiklik talebini inceler. Değişikliğin proje kapsamı, bütçe ve zaman çizelgesi üzerindeki etkilerini değerlendirir. Bu değerlendirme, değişikliğin kabul edilip edilmeyeceğine karar vermek için önemlidir.

Değişiklik Danışma Kurulu (Change Control Board - CCB): Bazı projelerde, değişikliklerin kabul edilip edilmemesine karar veren bir CCB bulunabilir. CCB, değişiklik taleplerini ayrıntılı bir şekilde gözden geçirir ve projenin amaçlarına uygunluğunu değerlendirir.

Proje Planının Güncellenmesi: Eğer bir değişiklik kabul edilirse, proje planı güncellenir. Bu, yeni kapsamın, bütçenin ve zaman çizelgesinin yansıtılması anlamına gelir.

Resmi Onay: Güncellenen proje planı, proje sponsoru veya yetkilisi tarafından resmi olarak onaylanmalıdır. Bu, değişikliğin projenin genel hedeflerine uygun olduğunu ve kaynakların uygun şekilde tahsis edildiğini doğrulamak için önemlidir.

1.2.2.3. Zaman Yönetimi

Zaman, bir projedeki temel kıstaslardan biridir. Zaman yönetimi; zaman çizelgesi yönetiminin planlanması, aktivitelerin tanımlanması, aktivitelerin sıralanması, kaynaklarının belirlenmesi, aktivite sürelerinin tahmin edilmesi, zaman çizelgesinin geliştirilmesi ve zaman çizelgesinin kontrolü süreçlerinden oluşur.

1. **Aktivitelerin Tanımlanması:** Bu aşamada Aktivite Listesi, Kilometre Taşı Listesi oluşturulur. Aktivitelerin tanımlanması süreci, proje yönetimi için temel bir adımdır ve proje aktivitelerinin tümünün listelenmesini içerir. Bu süreçte, proje sürecinde yapılması gereken tüm işleri belirleyip sıralayarak, her aktivitenin sorumluluklarını ve hedeflerini netleştiririz. Aktivite Listesi oluşturulurken, proje ekibinin ne zaman, hangi kaynaklarla, hangi işlerin yapılacağı belirlenir. Kilometre Taşı Listesi ise önemli dönüm noktalarını ve bu noktalar arasında geçen zamanı ifade eder. Bu liste, proje ilerledikçe gerçekleştirilen başlıca hedeflerin takibini kolaylaştırır. Başlangıç, ara ve bitiş noktaları belirlenerek, projedeki en kritik aktiviteler tanımlanır ve iş akışı planlanır.

2. **Aktivitelerin Sıralanması:** Belirlenen proje aktiviteleri arasındaki ilişkileri belirleme ve kayıt altına alma aşamasıdır. Aktivitelerin sıralanması, proje aktiviteleri arasındaki ilişkilerin net bir şekilde tanımlanmasını içerir. Bu süreç, her aktivitenin hangi sırayla yapılması gerektiğini belirler ve hangi aktivitelerin birbirine bağımlı olduğunu ortaya koyar. Bu sıralama, projenin zaman çizelgesinin oluşturulmasında önemli bir adımdır. Ayrıca, hangi aktivitelerin birbirine paralel olarak yapılabileceğini de gösterir, bu da kaynakların verimli kullanımına yardımcı olur. Aktivitelerin sıralanması için çeşitli teknikler ve araçlar kullanılabilir, örneğin ağ diyagramları (network diagrams) ile aktivitelerin sırası ve ilişkileri görsel olarak belirlenebilir.

3. **Aktivite Kaynaklarının Tahmin Edilmesi:** Her bir aktiviteyi gerçekleştirmek için gerekli malzeme, insan, araç ya da gereçlerin türünü ve miktarlarını tahmin etme aşamasıdır. Bu süreç, her bir aktivitenin gerçekleştirilmesi için hangi kaynakların gerektiğini belirler. Kaynaklar arasında insan gücü, malzeme, ekipman ve teknolojik gereçler yer alır. Bu kaynakların türü ve miktarı, her bir aktivitenin başarıyla tamamlanması için gereklidir. Kaynak tahminleri, proje ekibinin yeterli ve uygun kaynağa sahip olmasını sağlamak için önemlidir. Kaynakların doğru şekilde tahmin edilmesi, projede herhangi bir gecikme veya kaynak eksikliği yaşanmasını engeller. Ayrıca, kaynak maliyetlerini tahmin etmek, projenin bütçesini kontrol edebilmek açısından kritik rol oynar.

4. **Aktivite Sürelerinin Tahmin Edilmesi:** Belirlenen kaynaklarla her bir aktivitenin tamamlanması için gerekli çalışma sürelerinin belirlenmesi aşamasıdır. Belirleme tahmin etme teknikleri (Parametrik, 3 Nokta, Örneksel vb.) kullanılarak yapılır. Aktivite sürelerinin tahmin edilmesi, her bir aktivitenin tamamlanması için gereken süreyi belirleme sürecidir. Bu tahminler, daha sonra proje zaman çizelgesinin oluşturulmasında kullanılacak temel verilerdir. Süre tahminleri yaparken, çeşitli teknikler kullanılır; Parametrik tahminleme, 3 nokta tahminleme veya geçmiş projelerden elde edilen örnek verilerle yapılan tahminler gibi. Bu süreç, tahminin doğruluğunu artırmak için uzman

görüşlerinden ve benzer projelerden elde edilen deneyimlerden faydalanabilir. Ayrıca, her bir aktivite için tahmin edilen süreler göz önünde bulundurularak, iş akışındaki potansiyel engeller belirlenebilir ve zaman sıkıştırma gibi stratejiler uygulanabilir.

5. **Zaman Çizelgesinin Geliştirilmesi:** Proje zaman çizelgesi için, aktivite sıralamalarının, sürelerinin, kaynak gereksinimlerinin ve zaman çizelgesi kısıtlarının analiz edilme aşamasıdır. CPM, Kaynak Dengeleme, Zaman Sıkıştırma teknikleri kullanılarak yapılabilir. Zaman çizelgesinin geliştirilmesi süreci, projenin başlangıcından bitişine kadar olan tüm aktivitelerin sıralandığı, sürelerinin belirlendiği ve kaynak gereksinimlerinin göz önünde bulundurulduğu bir planlama sürecidir. Bu süreç, aktivitelerin birbirine olan ilişkilerini belirlemeyi ve projenin kritik yolunu (critical path) saptamayı içerir. Bu aşama, CPM (Critical Path Method), kaynak dengeleme (resource leveling) ve zaman sıkıştırma (crashing) gibi çeşitli teknikler kullanılarak yapılabilir. Zaman çizelgesinin geliştirilmesi, kaynakların verimli bir şekilde tahsis edilmesini sağlar ve projenin tüm aşamalarının zamanında tamamlanabilmesi için kritik bir temel oluşturur.

6. **Zaman Çizelgesinin Kontrolü:** Projedeki ilerlemeyi takip etmek ve güncellemek için projenin mevcut durumunun izlenmesi ve zaman çizelgesindeki değişiklerin yönetilmesi aşamasıdır. Varyans Analizi, Önden Gitme ve Beklemelerin yönetilmesi teknikleri uygulanır. Zaman çizelgesinin kontrolü, proje süresince planlanan aktivitelerin ilerlemesini izleme ve bu ilerlemenin zaman çizelgesi ile uyumlu olup olmadığını belirleme sürecidir. Bu süreçte, projede yapılan değişiklikler, aktivitelerin bitiş tarihleri ve gecikmeler göz önünde bulundurularak zaman çizelgesindeki güncellemeler yapılır. Varyans analizi, önden gitme ve beklemelerin yönetilmesi gibi tekniklerle projede yapılacak düzeltmeler ve iyileştirmeler planlanır. Projeye dair herhangi bir sapma belirlendiğinde, proje yöneticisi gerekli önlemleri alır ve zaman çizelgesini günceller.

Bu süreçler birbirleriyle ve diğer alanlardaki süreçlerle etkileşim halindedir. Her süreç, projenin ihtiyaçlarına bağlı olarak, çalışan sayısında değişkenlik gösterebilir. Her süreç her projede en az bir kez gerçekleştirilir ve proje fazlara ayrılıyorsa bir ya da birkaç fazda gerçekleştirilebilir.

Bir proje yöneticisinin veya ekibin zamanı etkili bir şekilde yönetmesi veya kötü yönetmesi, projenin sonucunu önemli ölçüde etkileyebilir. İşte projelerde etkin zaman yönetiminin ve kötü zaman yönetiminin meydana getireceği sonuçlar aşağıda ele alınmaktadır.

Etkin Zaman Yönetiminin Sonuçları:

Proje Başarısı: Etkin zaman yönetimi, projenin başarılı bir şekilde tamamlanmasına katkı sağlar. Görevler zamanında tamamlanır ve projenin son tarihine uyulur.

Daha Az Stres: İyi planlanmış projelerde, son dakika panikleri ve aşırı stres azalır. Ekip üyeleri, işlerini zamanında yapma konusunda daha rahat hissederler.

Daha İyi Kalite: Zamanın etkili bir şekilde kullanılması, işin daha iyi kalitede olmasına katkıda bulunur. Aceleye getirilmiş işlerin sayısı azalır ve hataların düzeltilmesi için daha fazla zaman ayrılabilir.

Maliyet Tasarrufu: Projelerde etkin zaman yönetimi, gereksiz maliyetlerin önlenmesine yardımcı olabilir. Zamanında tamamlanan görevler, bütçe aşımını engeller.

Daha İyi İş İlişkileri: Zamanında iletişim ve işbirliği, iş ilişkilerini güçlendirebilir. Paydaşlar, projenin ilerlemesi hakkında güncel bilgilere sahip olduklarında daha memnun olurlar.

Kötü Zaman Yönetiminin Sonuçları:

Proje Gecikmeleri: Kötü zaman yönetimi, projenin gecikmelere yol açmasına neden olabilir. Görevlerin sürekli olarak ertelenmesi veya son dakika panikleri projenin ilerlemesini engeller.

Stres ve Baskı: Kötü zaman yönetimi, ekip üyeleri üzerinde aşırı stres ve baskı oluşturabilir. Görevlerin son dakikada yetiştirilmesi, çalışanların iş memnuniyetini düşürebilir.

Kalite Sorunları: Zaman kısıtlamaları altında çalışmak, iş kalitesini olumsuz etkileyebilir. Hataların daha fazla olma olasılığı artar.

Maliyet Aşımı: Projenin zamanında tamamlanamaması, ek maliyetlere yol açabilir. Kaynakların verimsiz kullanılması nedeniyle bütçe aşımı riski artar.

İş İlişkilerinde Sorunlar: Kötü zaman yönetimi, paydaşlar ve müşteriler arasında güven kaybına yol açabilir. Projenin sürekli olarak ertelenmesi veya değiştirilmesi, iş ilişkilerini olumsuz etkileyebilir.

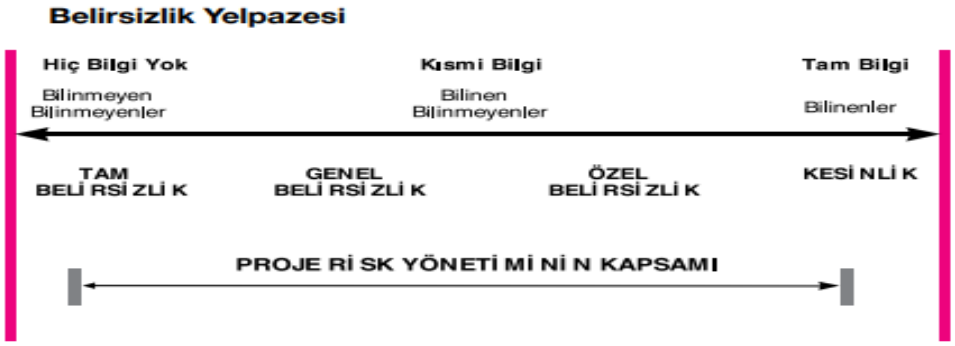
Sonuç olarak, projelerde etkin zaman yönetimi, başarı şansını artırırken kötü zaman yönetimi, projenin başarısızlığını tehdit edebilir. Proje yöneticileri ve ekipleri, zamanı etkili bir şekilde planlamak, önceliklendirmek ve takip etmek için çaba göstermelidirler. Bu, projenin başarısını ve tüm paydaşların memnuniyetini artırabilir.

1.2.2.4. Proje Risk Yönetimi

Proje risk yönetimi, istenmeyen olayların etkisinin/olasılığının kontrol altına alınması, izlenmesi ve en aza indirilmesi için kaynakların koordineli ve ekonomik uygulanabilmesi amacıyla risklerin tespiti, değerlendirilmesi ve önceliklendirilmesidir (Hubbard, 209). Proje risk yönetimi, risklerin tanımlanması, değerlendirilmesi ve yanıt verilmesi sürecinin hem teknik hem de yönetsel açıdan ele alır.

PMI'a göre, proje riskleri gerçekleştiklerinde; zaman, maliyet, kapsam veya kalite gibi en az bir proje hedefini olumsuz etkileyen belirsiz olay veya durumdur.

Geniş anlamda riske iki farklı yaklaşım söz konusudur: Birinci yaklaşımda risk, belirsizlik anlamına gelir. Bu durumda hem olumlu hem de olumsuz sonuçlar içermektedir. İkinci yaklaşımda risk tehdit/tehlike anlamına gelir. Bu durumda yalnızca olumsuz sonuçlar içerir. Risk, proje hedeflerini olumlu/olumsuz etkileyebilen belirsiz olayların yığılımlı etkisi olarak tanımlanmaktadır.



Belirsizlik Yelpazesi Şeması

Belirli Proje Riskleri

Belirli proje riskleri, kaynağı ve niteliklerine göre beş ana sınıfa ayrılır. Bu risklere aşağıda yer verilmektedir.

a) Dış kaynaklı, öngörülemeyen, kontrol edilemeyen;

- Mevzuat kaynaklı,
- Doğal afetler,
- Varsayılabilecek olaylar,
- Dolaylı etkiler,
- Tamamlanmama eksiklikleri.

b) Dış kaynaklı, öngörülebilen, kontrol edilemeyen;

- Pazar riskleri,
- İşletme,
- Çevresel etkiler,

- Sosyal etkiler,
- Parasal değişiklikler,
- Enflasyon,
- Vergilendirme.

c) İçten kaynaklanan, teknik olmayan, genellikle kontrol edilebilen;

- Yönetim,
- Program,
- Maliyet,
- Para akışı,
- Yeterlilik kaybı.

d) Teknik, genellikle kontrol edilebilen;

- Teknolojik değişiklikler,
- Performans,
- Projenin teknolojisine özgü risk,
- Tasarım,
- Projenin büyüklüğü ya da karmaşıklığı.

e) Hukuksal, genellikle kontrol edilebilen;

- Lisanslar/Patent hakları,
- Sözleşme,
- Örgüt dışından gelen hukuk davası,
- Örgüt içinden gelen hukuk davası,
- Mücbir sebepler

Risk ne zaman alınır?

Risk sadece, beklenen yararın başarısızlığın maliyetini ve kazanma şansını da kaybetme olasılığını aştığı durumlarda alınmalıdır. Risk alan aşağıdaki soruların gerçek cevabını aramalıdır:

Risk Niçin Alınmalıdır?

Bu soru, riskin neden alınması gerektiği hakkında bir anlayış geliştirmeyi amaçlar. Risk, genellikle belirli bir fırsatı yakalamak veya bir hedefi elde etmek için alınır. Bu nedenle, riskin neden gerekli veya arzu edilen bir sonuca ulaşmak için alındığını anlamak önemlidir.

Örnek: Bir teknoloji şirketi, mevcut ürünlerinin rekabetçiliğini sürdürmek ve büyümek için yeni bir pazar segmentine girmeye karar verir. Bu risk alınır, çünkü yeni pazarda büyüme potansiyeli ve rekabet avantajı bulunmaktadır.

Ne Kazanılacak?

Risk alındığında elde edilmesi beklenen fayda veya kazançlar net olarak tanımlanmalıdır. Bu, riskin amacını ve beklenen getiriyi netleştirir.

Örnek: Bir yatırımcı, yeni bir teknoloji girişimine yatırım yapmayı düşünüyor. Yatırımın başarılı olması durumunda, yatırımcıya yüksek bir getiri vaat ediliyor. Bu getiri, yatırımın potansiyel kazancını temsil eder.

Ne Kaybedilebilir?

Risk alındığında kaybedilmesi potansiyel olarak mümkün olan şeyler belirtilmelidir. Bu kayıplar finansal, itibari veya başka bir şekilde olabilir. Risk yönetimi, bu kayıpları minimize etmeyi veya kabul edilebilir seviyelere indirmeyi amaçlar.

Örnek: Bir işletme, yeni bir ürünü pazara sürerken, pazarın tepkisini ölçmek için büyük bir pazarlama kampanyası başlatır. Ancak kampanya beklenen ilgiyi çekmezse, işletme milyonlarca dolarlık maliyetle karşı karşıya kalabilir.

Başarı (ve Kaybetme) Şansı Nedir?

Risk alındığında, başarı veya başarısızlık olasılığı değerlendirilmelidir. Bu, riskin ne kadar güvenli veya riskli olduğunu belirlemeye yardımcı olur. Riskin başarı şansı yüksekse ve olası kayıplar düşükse, risk almak daha cazip olabilir.

Örnek: Bir spor takımı, bir önemli turnuvada şampiyonluk elde etmek için bir risk alır. Takımın başarı şansı, oyuncularının yetenekleri, antrenman programı ve rakip takımların gücü gibi faktörlere dayalı olarak değerlendirilir.

İstenen Sonuç Elde Edilemez İse Ne Yapılabilir?

Riskin başarısızlık durumunda nasıl ele alınacağı ve riskten kaçınma veya azaltma stratejileri düşünülmelidir. İstenen sonuç elde edilemezse, alternatif planlar veya düzeltici önlemler belirlemek önemlidir.

Örnek: Bir inşaat projesi, beklenen sürede tamamlanamazsa, proje yöneticileri alternatif tedarikçilerle çalışmayı veya ek kaynakları devreye almayı düşünerek proje sürecini hızlandırmaya çalışabilirler.

Beklenen Ödül Maruz Kalınacak Riske Değer Mi?

Riskin potansiyel ödülü, alınan riskle orantılı olmalıdır. Yani, riski almak için potansiyel ödül, riskin getirisiyle uyumlu olmalıdır. Eğer potansiyel ödül, riskin getirisini aşmıyorsa, risk alınmamalıdır.

Örnek: Bir yatırımcı, yüksek volatilite gösteren bir kripto para birimine yatırım yapmayı düşünüyor. Potansiyel kar, bu riski almayı cazip kılabilir ancak yatırımcı, olası kayıpları da dikkate almalıdır ve riskin getirisi ile uyumlu olup olmadığını değerlendirmelidir.

Risk ne zaman alınmaz?

Proje yöneticisinin risk almayacağı durumlar:

İşletme Bir Kayba Tahammül Edemediğinde: Eğer bir organizasyon veya proje, olası bir kaybı tolere edecek mali veya operasyonel kapasiteye sahip değilse, risk alınmamalıdır.

Örnek: Küçük bir yerel restoran, büyük bir tedarikçi fiyat artışı nedeniyle beklenmeyen maliyetlerle karşı karşıya kalabilir ve bu ek maliyetlere dayanacak kadar finansal kaynağa sahip değilse, bu riski almak mantıklı olmayabilir.

Maruz Kalınan Riskin Sonucu Çok Yüksek Olduğunda: Eğer riskin sonucu, organizasyon veya proje için çok yüksek bir mali veya itibari kayba yol açabilecek kadar büyükse, risk alınmamalıdır.

Örnek: Bir inşaat firması, çok yüksek maliyetli bir inşaat projesine girişmek istediğinde, potansiyel bir doğal afet veya ekonomik kriz gibi büyük bir riskin bu projeyi iflasa sürükleyebileceğini düşünmelidir.

Durum/Projenin Alınacak Riske Değmediği Durumlarda: Risk alınacak potansiyel ödül, organizasyon veya proje için uygun değilse, yani riskin getirisi riskin alınmasını haklı çıkarmıyorsa, risk alınmamalıdır.

Örnek: Bir yazılım şirketi, yeni bir ürün geliştirmek için yüksek bir riskli yatırım yapmayı düşünüyor ancak bu ürünün pazarda karşılayacağı talep ve kar getirisi çok düşükse, risk almak yerine başka projelere odaklanabilir.

Proje Lehinde Herhangi Bir Avantaj Söz Konusu Değilse: Eğer risk almak, projenin veya organizasyonun lehine herhangi bir avantaj sağlamıyorsa, risk alınmamalıdır.

Örnek: Bir enerji şirketi, yeni bir enerji üretim tesisi inşa etmek için yüksek bir çevresel risk taşıyorsa ve bu tesisin enerji üretiminde herhangi bir rekabet avantajı sağlamayacağını belirlediyse, bu riski almak mantıklı olmayabilir.

Yarışma Dürüst Olmayacaksa: Eğer bir organizasyon veya proje, yarışma sırasında dürüstlük ve etik kurallara uygun davranmayı riske atacaksa, risk alınmamalıdır.

Örnek: Bir inşaat firması, bir ihaleyi kazanabilmek için yarışmada rekabeti engellemeye veya yolsuzluk yapmaya yönelik bir riski almayı düşünüyorsa, etik kurallara uygun davranmalı ve bu riskten kaçınmalıdır.

Sağlanacak Yararlar Belirlenmemişse: Riskin potansiyel yararları net bir şekilde tanımlanmamışsa, risk alınmamalıdır.

Örnek: Bir yatırım şirketi, belirli bir finansal ürünün riskini almayı düşünüyor ancak bu ürünün potansiyel getirisi veya riskleri net bir şekilde tanımlanmamışsa, bu riski almak yerine daha fazla araştırma yapmalıdır.

Kabul Edilebilir Seçenek Sayısı Çok İse (Belirsizlikte Artar): Eğer organizasyon veya proje için uygun kabul edilebilir seçenekler çoksa ve risk almak gerekli değilse, risk alınmamalıdır. Özellikle belirsizliklerin fazla olduğu durumlarda risk alımı daha da azaltılmalıdır.

Örnek: Bir otomobil üreticisi, yeni bir modelin geliştirilmesi için birden fazla farklı yakıt türü veya motor seçeneği düşünüyorsa, çok sayıda seçenek arasından hangi seçeneği seçeceğini belirlemek için daha fazla analiz yapmalıdır.

Alınacak Risk Bir Proje Hedefine Ulaşmasında Yetersiz Kalıyorsa: Eğer risk alınacak olsa bile, sonuçta proje hedeflerine ulaşmada yetersiz kalıyorsa, risk alınmamalıdır.

Örnek: Bir inşaat projesi, mühendislik hesaplarının eksik veya hatalı olduğunu tespit ettiğinde, bu riski almak yerine mühendislik hesaplarını düzeltmeyi ve güvence altına almayı tercih etmelidir.

Varsayımların Beklenen Değeri Negatif İse (ya da Küçük Bir Değişiklik Negatif Yapıyorsa): Varsayımların veya girdilerin beklenen değeri negatifse veya küçük bir değişiklik negatif sonuçlara yol açıyorsa, risk alınmamalıdır.

Örnek: Bir finansal kurum, yeni bir yatırım yapmadan önce ekonomik varsayımların beklenen değerinin negatif olduğunu veya küçük bir değişikliğin olumsuz sonuçlara yol açabileceğini tespit ederse, bu riski almaktan kaçınmalıdır.

Eldeki Değerler Düzensiz Bir Dağılım Oluşturuyorsa: Değerlerin düzensiz bir şekilde dağıldığı durumlarda, risk almak daha dikkatli bir şekilde değerlendirilmelidir.

Örnek: Bir yatırım portföyü, bazı varlıkların değerinin büyük dalgalanmalar gösterdiği ve diğerlerinin stabil olduğu bir düzensiz dağılıma sahipse, risk yönetimi stratejileri, bu değişkenliği dikkate almalıdır.

Sonucu Hesaplamak İçin Yetersiz Veri Yoksa: Riskin sonucunu hesaplamak için yeterli ve güvenilir veri yoksa risk alınmamalıdır.

Örnek: Bir biyomedikal araştırma projesi, eldeki verilerin yetersiz olduğunu ve proje sonuçlarının hesaplanamayacağını belirlerse, projenin devam etmesi riskli olabilir ve daha fazla veri toplama çabası gerekebilir.

Sonucun Tatminkar Olmaması Halinde Uygulanacak Bir Olasılık/Maliyet Planı Yoksa: Eğer riskin sonucu tatminkar değilse ve bu sonucu düzeltecek bir olasılık veya maliyet planı yoksa, risk alınmamalıdır.

Örnek: Bir inşaat projesi, projenin sonucunun tatmin edici olmadığını ve maliyetlerin arttığını belirlediğinde, bu duruma nasıl müdahale edileceği veya maliyetlerin nasıl kontrol altına alınacağı konusunda bir planı yoksa, risk alınmamalıdır.

Risk yönetimi, projenin amacına ulaşmasına yardımcı olması için bir projenin yaşam döngüsü boyunca ortaya çıkabilecek potansiyel ve beklenmedik sorunların belirlenmesi ve yönetilmesi çalışma

çabalarını ifade etmektedir. Projelerde yaşanan problemlerin %90'ı öngörülebilmektedir. Reaktif yaklaşım (problem olduğunda müdahale) kayıpların yaşanmasına sebep olabilmektedir. Proaktif yaklaşım olan proje risk yönetimi, kayıpların önlenmesi veya olumsuz etkilerini en aza indirilmesini sağlamaktadır.

Proje risk yönetiminin hedefleri projede olumlu olayların olasılık ve etkilerinin arttırılması, olumsuz olayların olasılık ve etkilerinin azaltılmasıdır. Risk yönetimi, farklı proje türlerinde farklı anlamlara gelebilmektedir. Büyük ölçekli projelerde, risk yönetimi stratejileri, sorunlar ortaya çıktığında azaltma stratejilerinin uygulanmasını sağlamak için her risk için kapsamlı ayrıntılı planlama içerebilmektedir. Daha küçük projeler için risk yönetimi, yüksek, orta ve düşük öncelikli risklerin basit, önceliklendirilmiş bir listesi anlamına gelebilmektedir.

Başarılı bir risk yönetiminin anahtarı erken tanımlama, planlama ve kararlı bir uygulamadır. İyi planlama; kapsamlı ve yinelenen bir yaklaşımla risk tanımlama, değerlendirme ve tepki geliştirmeyi mümkün kılar. Risk yönetimi başlı başına bir yönetim disiplinidir, ancak belirsizlikleri ve riskleri tamamen ortadan kaldıracak sihirli bir yönetim disiplini değil, potansiyel risklerin sistematik olarak değerlendirilerek, olası zararlarının etkisini azaltıcı yönde, verilere dayalı karar vermeyi sağlayan bir disiplindir ve diğer disiplinlerle bir bütünlük içerisinde uygulanması gerekir (Özkılıç, 2014).

Risk yönetimi ikilemi grafik modelinde gerçekleşen bir risk olayındaki değişiklikler (zaman tahminlemesindeki, maliyet kestirimindeki veya dizayn teknolojisindeki bir hata), projenin konsepti, planlaması ve başlangıç evresi açısından çok önemlidir. Projede bir risk olayının maliyete etkisi, risk olayı ne kadar geç meydana gelirse o kadar çoktur. Projenin başlangıç safhaları potansiyel risk olaylarının belirlenmesi ve etkilerinin en küçüklenmesi için en elverişli safhalardır. Proje yarılandıktan sonra meydana gelen risk olaylarının maliyeti de hızla artar. Projenin risk olaylarını proje başlamadan önce belirlemek ve bu muhtemel risklerin karşılıklarını hazırlamak, bu risk olaylarını proje süreci içinde yönetmekten daha ihtiyatlı ve mantıklı olacaktır.

Risk yönetimi reaksiyonel bir yaklaşım değil, riske mantıklı yanıt geliştiren bir yaklaşımdır. Aslında risk yönetimi, sürprizleri azaltmayı ve istenmeyen olayların olumsuz sonuçlarını minimize etmeyi sağlayan bir önleyici süreçtir. Risk yönetimi, proje yöneticisine gerektiğinde zaman, maliyet veya teknik avantajlar söz konusu olduğunda risk alabilme olanağı vermektedir. Proje risklerinin başarılı yönetimi, proje yöneticisine gelecek üzerinde daha iyi bir kontrol imkânı vermekte ve projenin amaçlarına zamanında, belirlenen bütçe ile ve teknik gereklilikleri karşılayacak şekilde ulaşma şansını önemli ölçüde artırmaktadır.

Proje riskinin iki ana kategorisi vardır:

1- İş yararlarını etkileyen kategori,

2- Projenin kendisini etkileyen kategori.

Proje sponsoru ilk risk kategorisini hafifletmekten, proje yöneticisi ise ikinci kategoriyi hafifletmekten sorumludur.

Proje riskine ilişkin adımlara aşağıda yer verilmektedir.

Adım 1 Riskin Belirlenmesi:

Risk yönetimi süreci proje üzerinde etkili olabilecek tüm olası risklerin bir listesinin oluşturulmaya çalışılması ile başlamaktadır. Tipik olarak planlama süreci boyunca proje yöneticisi, risk yönetimi takımını ve diğer ilgilileri bir araya getirmektedir. Bir araya gelen takım beyin fırtınası ve diğer problem belirleme teknikleri ile potansiyel problemleri belirlemektedir. Değerlendirme süreci boyunca takım üyeleri belirlenen riskleri analiz etme ve uygun olmayanları eleme şansı bulmaktadırlar. Risk belirleme sürecinin ön safhalarında çok sık karşılaşılan bir hata, olabilecek olayların sonuçlarından çok olmuş olayların sonuçları üzerinde odaklanmaktır. Örneğin takım üyeleri bu yüzden projedeki asıl riski gözden kaçırabilirler. Asıl üzerinde odaklanılması gereken riskin gerçekleşmesine neden olabilecek olaylardır.

Başlangıçtaki odak noktası, projenin spesifik noktalarındaki risklerden çok projenin tamamı üzerinde etkili olan risklerdir. Makro riskler tanımlandıktan sonra projenin spesifik noktalarındaki

riskler ele alınabilmektedir. Mikro risklerin belirlenmesi için etkili bir araç, iş kırılım yapısıdır. İş kırılım yapısının kullanımı risklerin gözden kaçırılma şansını azaltmaktadır. Büyük projelerde spesifik noktalarda çoklu risk takımları organize edilmekte ve bunların raporları toplanarak proje yöneticisine iletilmektedir.

Risk profili, yöneticilerin riski tanımlamaları ve analiz etmeleri için diğer bir araçtır. Risk profili, geleneksel olarak proje üzerindeki belirsizlikleri arayan bir soru listesidir. Bu sorular daha önceki benzer projelerden elde edilen deneyimler yardımıyla belirlenmektedir. İyi bir risk profili, ele alınan projenin tipine göre yeniden biçimlendirilmektedir. Risk profilleri firmanın yalnızca güçlü ve zayıf yönlerini ortaya koyar. Sonuç olarak risk profilleri, firmanın hem teknik hem yönetsel riskleri belirlemektedir.

Risk belirleme süreci çekirdek kadro ile sınırlandırılmamalıdır. Müşterilerden, sponsorlardan, taşeronlardan, satıcılardan ve diğer katılımcılardan sürekli bilgi istenmelidir. İlgili katılımcılarla resmi olarak görüşülmekte veya risk yönetimi takımına alınabilmektedir. Amaç olayları meydana gelmeden önce tespit etmektir. İş kırılım yapısı ve risk profilleri, ele alınmamış olay kalmadığından emin olmak için önemli iki araçtır. Aynı zamanda, iyi yapılmış bir risk tanımlama işlemi ile risk bir miktar önlenebilmektedir. Proje yöneticisinin doğru bir tavır alması ve yönetim sürecini tamamlaması önemlidir.

Risk belirlemede yardımcı olabilecek bazı sorular aşağıdaki gibidir:

- Amaca ulaşma yolunda neler yanlış gidebilir?

Örnek: Bir işletme, yeni bir ürünü piyasaya sürmek istediğinde, pazarlama stratejilerinin etkisiz olması veya rekabetin daha güçlü olması nedeniyle hedeflere ulaşmakta zorluk yaşayabilir.

Nerelerde Sorun Yaşanır?

Örnek: Bir otel, misafir memnuniyetini artırmak için hizmet kalitesini yükseltmeye çalışırken, personel eğitim eksikliği nedeniyle hizmetlerde sorunlar yaşanabilir.

- Başarısız olmaya neden olabilecek işler nelerdir?

Örnek: Bir yazılım şirketi, yeni bir uygulama geliştirirken, yazılım hataları veya kullanıcı dostu olmayan bir arayüz nedeniyle kullanıcıların uygulamayı terk etmesi riskiyle karşı karşıya olabilir.

- Hangi alanlarda ya da yerlerde zayıflıklar var?

Örnek: Bir hükümet kurumu, siber güvenlik kontrollerinin eksik olduğu belirli sistemlerin tespit edilmesi sonucunda siber saldırı riskini azaltmak için önlemler alabilir.

- Hangi varlıklar daha çok korunmalı?

Örnek: Bir sanat galerisi, nadir ve değerli sanat eserlerini daha sıkı güvenlik önlemleri altında tutarak hırsızlık riskini azaltabilir.

- Hırsızlık veya yolsuzluk alanları neler olabilir?

Örnek: Bir kamu kurumu, ihale süreçlerinde yolsuzluk yapma potansiyeli taşıyan belirli bölgeleri tespit ederek bu alandaki denetimleri artırabilir.

- Faaliyetler hangi durum ya da olaylar karşısında aksayabilir, durabilir?

Örnek: Bir enerji şirketi, enerji kaynaklarının kesilmesi veya enerji üretim tesislerinde ciddi arızalar durumunda enerji üretiminin durma riskiyle karşı karşıya olabilir.

- En kritik bilgi kaynakları nelerdir?

Örnek: Bir hukuk firması, müşteri bilgilerinin gizliliğini korumak için müşteri dosyalarını en üst düzeyde güvenlik altına alır.

- En fazla harcama yapılan alanlar hangileridir?

Örnek: Bir inşaat projesinde, ana yüklenici tarafından malzeme alımlarının ve alt yüklenici sözleşmelerinin dikkatle yönetilmesi, bütçe kontrolünü sağlamak için kritiktir.

- Takdire dayanan, bir kişi tarafından alınan kritik kararlar nelerdir?

Örnek: Bir teknoloji şirketi, üst düzey yöneticilerin yeni bir ürünün piyasaya sürülmesi veya geliştirilmesi konusundaki kritik kararlarını dikkatle değerlendirir.

- Hangi faaliyet veya süreçler daha karmaşıktır?

Örnek: Bir uzay ajansı, uzay misyonlarını planlarken ve yürütürken karmaşık roket fırlatma ve uzay aracı yönetimi süreçleriyle karşı karşıya olabilir.

- İdari, mali ve cezai yaptırımlara maruz kalınan alanlar hangileridir?

Örnek: Bir finansal kurum, düzenleyici uyumluluk gereksinimlerini karşılamada başarısızlık durumunda mali cezalara ve yasal yaptırımlara maruz kalma riskini yönetmek için iç denetimleri güçlendirir.

Adım 2 Risk Ölçme:

Adım 1 ile potansiyel risklerin bir listesi oluşturulmaktadır. Ancak bu risklerin tümü dikkate değer değildir. Bazı riskler aşikar ve görmezden gelinebilecek boyuttayken, bazıları projenin başarısı açısından çok ciddi olabilmektedir. Bu aşamada proje yöneticisi, listeden önemsiz ve gereğinden fazla önem verilmiş olan riskleri elemek için bir yöntem geliştirmelidir.

Senaryo analizleri, risk analizlerinde en çok kullanılan ve en kolay uygulanan yöntemdir. Takım üyeleri her bir riski aşağıdaki durumlar açısından değerlendirmelidir.

- Kabul edilemez olaylar,

- Olayların meydana gelişlerinin tüm sonuçları,

- Olayların etkilerinin büyüklüğü ve şiddeti,

- Olayların meydana gelme olasılıkları,

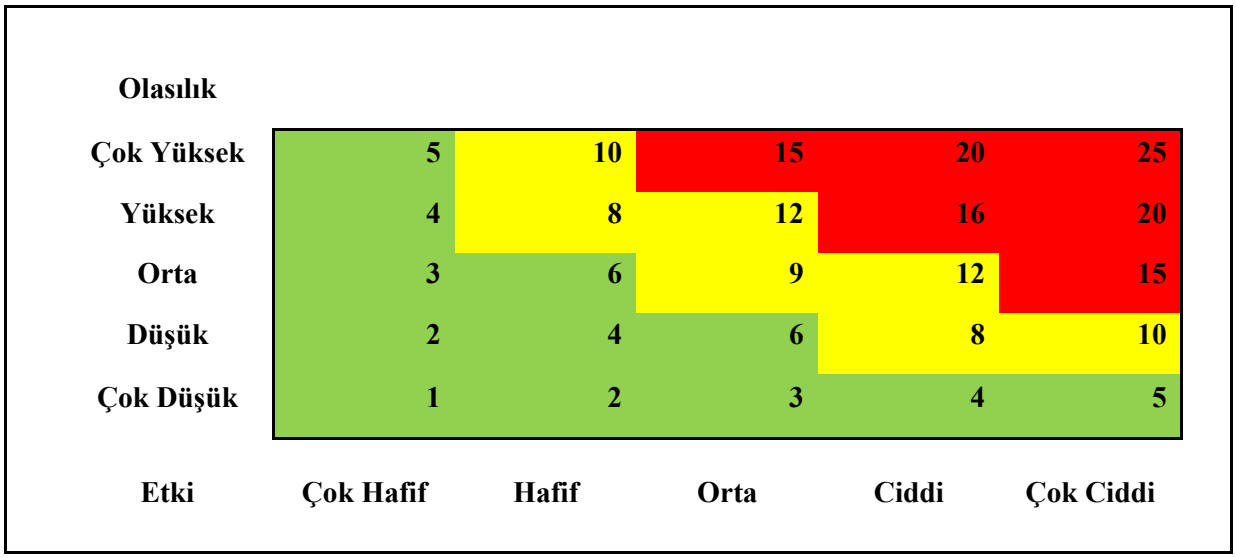
- Olay projenin hangi aşamasında meydan gelebileceği,

- Ele alınan projenin diğer kısımları veya başka projelerle olan etkileşimi.

Örneğin belirli bir materyalin temininde sıkıntı yaşanması olasılığı yüzde 80 olsun. Bu sıkıntının yaşanması; projenin ertelenmesi, sıkı program, düşük esneklik ve maliyet artışı gibi sonuçlara yol açabilir. Olayın etkisi ile maliyet %10 ve proje süresi %5 artabilir. Bu sıkıntı, projenin dizayn aşamasında meydana gelecektir. Bu projenin gecikmesi belki de başka projeleri de geciktirecek veya önceliklerini değiştirecektir. Bu bilginin elde edilebilmesi riskin ölçülmesini kolaylaştıracaktır.

İşletmeler genellikle farklı risk büyüklüklerini tek bir risk ölçüm matrisinde kategorize etmeyi kullanışlı bulmaktadırlar. Matris tipik olarak risk olaylarının olabilirlikleri ve olumsuz etkilerini dikkate almaktadır.

Risk şiddet matrisinde hücreler sırasıyla büyük, orta ve düşük riskleri ifade edecek şekilde kırmızı, yeşil ve sarı renkli bölgelere ayrılmaktadır. Kırmızı bölge matrisin sağ üst köşesine, yeşil bölge sol alt köşesinde, sarı bölge ise orta bölgede yer almaktadır. Riskin etkisi olabilirliğinden daha önemli kabul edildiği için kırmızı bölge olabilirlik ekseninde en aşağılara kadar inmektedir. Risk şiddet matrisi riskler arasında bir öncelik sıralaması oluşturmak için bir temel oluşturmaktadır. Birinci öncelik kırmızı bölgedeki risklere ikinci öncelik sarı bölgedeki risklere ayrılmaktadır. Yeşil bölgedeki riskler ise en son ele alınmaktadır.



Risk Şiddet Matrisi

Etki x Olasılık x Belirlenememe = Risk Değeri

Yukarıdaki formüldeki her üç bileşen de beş üzerinden ölçüme göre derecelendirilir. Örneğin, belirleme (detection), proje takımının risk olayının ne kadar yakında gerçekleşeceğinin farkına varabilme becerisi olarak tanımlanır. Bu bağlamda, 1 puan bir şempanzenin bile yaklaştığını kolaylıkla anlayabileceği türden risklere verilir. En yüksek puan 5 ise gerçekleşeceği fark edildiğinde artık çok geç denecek türden risklere verilir (örneğin sistem freezing). Benzer bir derecelendirme olayların meydana gelme olasılıklarına ve etkilerine de uygulanabilir. Risklerin ağırlıklandırılması tüm riskler için hesaplanacak olan bu "*risk değerl*eri" temel alınarak yapılır. Örneğin 1 etki puanı olan, çok düşük olasılıklı ve önceden belirlenebilmesi çok kolay olan bir olay için risk değeri 1 (1x1x1=1) olacaktır. Diğer taraftan, etki puanı 5, meydana gelme olasılığı çok yüksek ve önceden belirlenebilmesi imkânsıza yakın bir olayın risk değeri 125 (5x5x5=125) olur. Risk değerleri için belirlenen bu 1-125 aralığı risk olaylarını sınıflandırmaya yardımcı olur.

Adım 2.1 Olasılık Analizi:

Proje riskinin ölçülmesi sürecinde proje yöneticilerinin yararlanabileceği birçok istatistiksel yöntem vardır. Karar ağaçları, alternatif eylem biçimlerini beklenen değerlerini dikkate alarak değerlendirir. Net bugünkü değerin (NPV) istatistiksel değişimi, projedeki nakit akışında karşılaşılabilecek risklerin ölçümünde kullanılır. Geçmiş projelerin nakit akışları ile *S*-eğrileri (*S*-curves) (proje süresince herhangi bir andaki toplam maliyeti gösteren eğri) arasındaki korelasyon projenin nakit akış riskinin belirlenmesinde kullanılır.

PERT ve PERT simülasyon yapılan işleri gözden geçirmek ve proje riskini ölçmekte kullanılır. PERT ve diğer benzer teknikler projedeki tüm maliyetleri dikkate alarak daha makro açıdan riskleri sıralar. Burada tek tek olaylar üzerinde değil, projenin belirlenen zamanda ve belirlenen bütçe ile tamamlanabilmesinin olabilirliği üzerinde odaklanılacaktır. Bu teknikler projenin tüm risklerini ve gerekli olan ek sermaye, kaynak ve zamanın ölçülmesinde oldukça kullanışlıdır. PERT simülasyonun kullanımı gittikçe artmaktadır. Çünkü PERT için gerekli olan aynı verileri kullanır ve simülasyonu gerçekleştirmek için gerekli yazılımlar mevcuttur.

Yukarıda bahsedilen teknik kavramlar aşağıda daha detaylı olarak maddeler halinde açıklanmıştır:

Karar Ağaçları:

Karar ağaçları, proje yöneticilerine farklı eylem seçeneklerini ve bu seçeneklerin beklenen değerlerini analiz etmelerine yardımcı olur. Bu yöntem, karar verme süreçlerini görselleştirmek ve alternatifler arasında en iyi seçeneği belirlemek için kullanılır.

Avantajlar:

Karar ağaçları, karmaşık karar verme süreçlerini görselleştirir ve alternatif seçeneklerin sonuçlarını net bir şekilde ortaya koyar.

Beklenen değerler kullanılarak olası sonuçların değeri hesaplanabilir, bu da daha iyi kararlar alınmasını sağlar.

Karar ağaçları, farklı senaryoları göz önüne alarak risk analizi yapma yeteneği sunar.

Dezavantajlar:

Karar ağaçlarının oluşturulması ve yönetilmesi zaman alıcı olabilir.

Karar ağaçları, gelecekteki olayları ve sonuçları kesin bir şekilde tahmin edemez, bu nedenle belirsizlik durumlarına karşı kısıtlıdır.

Net Bugünkü Değer (NPV) Analizi:

NPV analizi, proje maliyetlerini ve getirilerini dikkate alarak projenin bugünkü değerini hesaplar. İstatistiksel değişiklikler, projenin nakit akışındaki belirsizlikleri ve riskleri değerlendirmek için kullanılır. NPV analizi, projenin finansal performansını anlamak için önemlidir.

Avantajlar:

NPV analizi, proje getirisini bugünkü değer olarak ifade ederek farklı projeleri mukayese etmeyi sağlar.

İstatistiksel değişikliklerin dikkate alınması, projenin finansal risklerini daha iyi anlamayı sağlar.

Dezavantajlar:

NPV analizi, gelecekteki nakit akışlarının tahminine dayalıdır, bu nedenle tahminlerdeki hatalar riski artırabilir.

İndirgeme oranı (diskont oran) seçimi konusundaki belirsizlik, sonuçları etkileyebilir.

***S*-eğrileri (*S*-Curves):**

S-eğrileri, projenin belirli bir anında toplam maliyetini gösteren bir eğridir. Bu eğri, projenin ilerlemesiyle maliyetlerin nasıl değiştiğini gösterir. *S*-eğrileri, proje maliyetlerinin zaman içindeki dağılımını ve risklerini anlamak için kullanılır.

Avantajlar:

S-eğrileri, projenin maliyetlerinin zaman içindeki dağılımını görselleştirir ve maliyetlerin kontrol edilmesine yardımcı olur.

Riskli alanları belirlemek için kullanışlıdır ve proje yönetimini iyileştirmeye yardımcı olabilir.

Dezavantajlar:

S-eğrileri, maliyetlerin tahmini konusunda bazı varsayımlara dayalıdır, bu nedenle gerçek dünya koşullarını tam olarak yansıtmayabilir.

Değişkenliklerin nedenlerini açıkça ortaya koymaz.

PERT (Program Evaluation and Review Technique):

PERT, proje süreçlerini ve aktivitelerini analiz etmek ve zaman tahminleri yapmak için kullanılan bir tekniktir. PERT analizi, belirsizlik ve riskleri dikkate alarak projenin süresini ve zaman çizelgesini belirlemeye yardımcı olur.

Avantajlar:

PERT, projenin süresini ve zaman çizelgesini tahmin etmek için kullanışlıdır.

Belirsizlikleri ve riskleri dikkate alarak projenin zaman yönetimini geliştirir.

Dezavantajlar:

PERT, projenin tamamlanma zamanı hakkında kesin bir tahmin vermez, olası bir zaman aralığı sunar.

Tahminler, projenin karmaşıklığına ve belirsizliğine bağlı olarak değişebilir.

PERT Simülasyon:

PERT simülasyonu, projenin belirsizliklerini ve risklerini daha ayrıntılı bir şekilde değerlendirmek için kullanılır. Bu yöntem, projenin farklı senaryolarını simüle ederek olası sonuçları analiz etmeye olanak tanır.

Avantajlar:

PERT simülasyonu, projenin farklı senaryolarını simüle ederek risk analizi yapma yeteneği sunar.

Proje risklerini daha ayrıntılı bir şekilde değerlendirmeyi sağlar.

Dezavantajlar:

Simülasyon yapmak için bazı verilere ve yazılımlara ihtiyaç vardır, bu da ek kaynak ve zaman gerektirebilir.

Sonuçlar, kullanılan girdilere ve senaryolara bağlı olarak değişebilir.

Adım 2.2 Yarı Nicel Analiz:

Proje yöneticileri risk analizi için olasılık üretilmesinde ve kullanılmasında çoğu zaman isteksizlerdir. Çünkü proje takımına riski telaffuz ettirmektir. Bu bilgi çok pratik olabilir ve aynı zamanda, olasılık ve fayda (utility) teorisinin yararlarını da beraberinde getirmektedir.

Proje yöneticileri tarafından kullanılan yaklaşımlardan biri yarı nicel senaryo analizidir. Birçok risk türünün zamana bağımlı olması, projenin ertelenmesini etkilemesi ve risk takımı üyeleri tarafından kolayca anlaşılması nedeniyle bu yaklaşım süreleri kullanılmaktadır.

Yaklaşık olarak nicel senaryo yaklaşımı, bir sonraki adım için senaryo analizini temel almaktadır. Etkilerin doğrulanmasında numaraların kullanılması tanımlanan risklerin ve analizlerinin kontrolünde bir gerçeklik sunmaktadır.

Özellikler ve Avantajlar:

Risk Bilincini Artırır: Yarı Nicel Analiz, proje takımına risk kavramını daha anlaşılır hale getirir ve projedeki potansiyel riskleri daha belirgin bir şekilde görünür kılar. Bu, riskleri projenin her aşamasında daha iyi anlamalarını sağlar.

Zamana Bağlı Riskleri Ele Alır: Birçok proje riski zamanla değişebilir veya evrilebilir. Yarı Nicel Analiz, zamanın etkisini dikkate alarak risklerin nasıl değişebileceğini gösterir ve bu, projenin zaman çizelgesinin yönetilmesine yardımcı olur.

Risk Takımı İçin Eğitici: Bu yaklaşım, risk takımının projedeki riskleri daha iyi anlamasına yardımcı olur ve risklerin etkilerini sayısal olarak gösterir. Bu, risk takımının riskleri daha etkili bir şekilde ele almasına katkı sağlar.

Senaryo Analizi Temel Alır: Yarı Nicel Analiz, projenin farklı senaryolarını ele alır. Bu, projenin farklı koşullar altında nasıl performans gösterebileceğini ve hangi risklerin hangi senaryolarda daha belirgin olduğunu gösterir.

Kontrol ve Önlemler için Temel Sağlar: Yarı Nicel Analiz, riskleri sayısal verilerle değerlendirdiği için risklerin kontrol altına alınması ve önlemlerin alınması için somut bir temel sunar.

Karar Alma Sürecine Katkı Sağlar: Bu yaklaşım, proje yöneticilerinin ve paydaşların daha bilinçli kararlar almasına yardımcı olur. Risklerin sayısal olarak ifade edilmesi, alternatif eylemler arasında daha iyi kararlar verilmesine yardımcı olur.

Gerçekçi Bir Perspektif Sunar: Yarı Nicel Analiz, projenin gerçek dünyadaki risklerini daha iyi yansıtabilir. Sayısal veriler, risklerin daha net bir şekilde anlaşılmasını sağlar.

Yarı Nicel Analiz, proje yöneticileri için risk analizi sürecini daha yapılandırılmış ve anlaması daha kolay hale getiren bir yaklaşım sağlar. Ancak, her projenin kendine özgü ihtiyaçları ve koşulları vardır, bu nedenle hangi risk analizi yönteminin kullanılacağı dikkatlice değerlendirilmelidir.

Adım 3 Riske Tepki Geliştirme:

Bir risk olayı tanımlandığında ve değerlendirildiğinde, bu olay için hangi tepkinin uygun olduğuna kararı verilmelidir. Riske verilecek tepkiler; riski azaltma, riskten kaçınma, riski transfer etme, riski paylaştırma ve riski önleme gibi sınıflara ayrılabilmektedir.

Adım 3.1 Riskin Azaltılması:

Riskin azaltılması, ilk olarak dikkate alınan ve potansiyel kayıpları azaltmakla ilgili alternatiftir. Riskin azaltılması için iki strateji vardır. Bunlar risk olayının meydana gelişinin olabilirliğinin azaltılması ve/veya risk olayının proje üzerindeki olumsuz etkisinin azaltılmasıdır. İlk strateji için bir bilgi sistemi projesi örnek olabilir. Konu proje süresinin ve maliyetinin gerçek değerinin altında kestirilmesi olduğundan proje yöneticileri bu belirsizlikleri telafi etmek için kestirimlerini arttırmaktadırlar. Proje süresinin ve maliyetinin ayarlanabilmesi için geçmiş benzer bir proje ile ele alınan proje arasındaki oran oldukça sık kullanılır. Oran tipik olarak sabittir. Örneğin, eğer eski bir projede her bir satır bilgisayar programının kodunun yazılması 10 dakika sürmüşse, 1.10 oranı yeni projede bu sürenin 11 dakika olacağı anlamına gelmektedir.

Risk azaltma stratejileri oldukça önemli ve yaygın olarak kullanılan yaklaşımlardır. Bu kapsamda ele alınabilecek stratejilerin ayrıntılarına ve örneklerine aşağıda yer verilmiştir:

Risk Olayının Olasılığının Azaltılması:

Bu strateji, risk olayının gerçekleşme olasılığını azaltmayı amaçlar. Riskin gerçekleşme olasılığını azaltmak, proje yöneticilerinin daha güvenli bir projeye sahip olmalarını sağlar.

Örnek: Bir inşaat projesinde, işçi güvenliği riskini azaltmak için daha fazla eğitim verilebilir ve işçilerin güvenlik önlemlerine daha sıkı bir şekilde uymaları teşvik edilebilir. Bu, iş kazalarının olasılığını azaltabilir.

Risk Olayının Etkisinin Azaltılması:

Bu strateji, risk olayının gerçekleşmesi durumunda projenin üzerindeki olumsuz etkisini azaltmayı amaçlar. Bu, riskin gerçekleşmesi durumunda zararın minimize edilmesine yardımcı olur.

Örnek: Bir yazılım projesinde, yazılım hatalarının neden olabileceği olumsuz etkileri azaltmak için düzenli olarak kod incelemeleri yapılabilir ve hata düzeltme süreçleri hızlandırılabilir. Bu, olası hataların proje üzerindeki etkisini azaltabilir.

Geçmiş Verilere Dayalı Kestirimler:

Bu strateji, projedeki belirsizlikleri ve riskleri azaltmak için geçmiş benzer projelerin verilerine dayalı tahminler kullanmayı içerir. Bu, proje süresi, maliyeti ve diğer faktörler hakkında daha kesin bilgilere sahip olmayı sağlar.

Örnek: Bir yazılım geliştirme projesi için proje süresini tahmin etmek için, geçmişte benzer projelerde elde edilen süre verileri kullanılabilir. Örneğin, bir benzer projede her bir yazılım modülünün geliştirme süresi ortalama 10 gün ise, bu bilgi yeni projede kullanılabilir.

Oranlarla Ayarlama:

Bu strateji, eski projelerden elde edilen oranları kullanarak belirsiz tahminleri ayarlamayı içerir. Bu oranlar, belirli bir değişkenin (örneğin, süre veya maliyet) eski projelerdeki değeri ile yeni projedeki tahmin değeri arasındaki ilişkiyi temsil eder.

Örnek: Geçmiş bir yazılım projesinde, her bir yazılım modülünün geliştirme süresi, kod satırı başına 10 dakika olarak hesaplanmışsa ve yeni projede benzer bir modülün 1000 kod satırı içerdiği tahmin ediliyorsa, bu bilgiye dayalı olarak yeni projenin geliştirme süresi 10,000 dakika (yaklaşık 6.94 gün) olarak ayarlanabilir.

Adım 3.2 Riskten Kaçınma:

Riskten kaçınma, riski ya da etkilerini tamamen ortadan kaldırmaya yönelik geliştirilen stratejidir. Riskin ortadan kaldırılması amacıyla proje planının değiştirilebilir. Bu yöntemle tüm risklerin ortadan kaldırılması mümkün olmasa da, proje başlamadan önce bazı belirli risklerden kurtulunabilir. Örneğin, deneme aşamasında olanlar yerine doğruluğu ispatlanmış teknolojilerin kullanılması teknik riskleri ortadan kaldıracaktır. Endonezyalı bir tedarikçi yerine Avustralyalı bir tedarikçinin seçilmesi kritik materyallerin sağlanması konusunda karşılaşılabilecek politika kaynaklı riskleri ortadan kaldırabilir.

Bazı riskler kaçınılmazdır. Örneğin, iflas riski, takım sorumluluğu riski ya da işletmeler ve insanlar için erken ölüm riski kaçınılmazdır. Bu kaybetme riski genellikle azaltılabilir ama yok edilemez. Kaçınmak diğer riskler için yalnızca kabul edilebilir bir alternatiftir. Temel kural; kaybetme riski yüksekse ve kaybetme şiddetti de yüksekse kaçınmak çoğu kez en iyisidir ve bazen tek pratik seçenektir.

Riskten kaçınma stratejileri oldukça önemli ve yaygın olarak kullanılan yaklaşımlardır. Bu kapsamda ele alınabilecek stratejilerin ayrıntılarına ve örneklerine aşağıda yer verilmiştir:

Riskleri Ortadan Kaldırma:

Bu strateji, projenin başlangıcında belirli riskleri tamamen ortadan kaldırmayı hedefler. Bunun için proje planı veya stratejisi değiştirilebilir.

Örnek: Bir yazılım geliştirme projesinde, belirli bir yazılım bileşeni geliştirilirken ortaya çıkan teknik riskleri ortadan kaldırmak için farklı bir yazılım teknolojisi veya daha güvenilir bir yaklaşım seçilebilir.

Tecrübeli ve Güvenilir Kaynakları Kullanma:

Bu strateji, riski azaltmak veya ortadan kaldırmak için daha önce başarılı sonuçlar veren ve güvenilir olarak bilinen kaynakları tercih etmeyi içerir.

Örnek: Bir inşaat projesinde, daha önce birlikte çalışılan ve güvenilir bir inşaat firması seçilerek projenin işlerlik riski azaltılabilir.

Politika ve İdari Kararlara Bağlı Riskleri Ortadan Kaldırma:

Bu strateji, projenin dışsal faktörlerle ilgili riskleri azaltmayı amaçlar. Bu, politika veya yönetmeliklere bağlı riskleri içerebilir.

Örnek: İhracat yapan bir şirket, siyasi istikrarsızlık nedeniyle belirli bir ülkeye satış yapma riskini ortadan kaldırmak için farklı bir pazar hedefleyebilir.

Kritik Tedarikçileri Seçme:

Bu strateji, projede kritik öneme sahip olan tedarikçilerin dikkatlice seçilmesini içerir. Güvenilir ve istikrarlı tedarikçilerle çalışmak, tedarikçi kaynaklı riskleri azaltabilir.

Örnek: Bir otomobil üreticisi, kritik bileşenlerin sağlanması için güvenilir ve uzun süreli ilişkiler kurduğu tedarikçileri seçerek tedarikçi kaynaklı riskleri minimize edebilir.

Ölçülebilir ve Yönetilebilir Riskler İçin Alternatifler Üretme:

Bu strateji, ölçülebilir ve yönetilebilir riskler için alternatif planlar veya yaklaşımlar geliştirmeyi içerir. Bu alternatifler, riskin etkilerini minimize etmeye yardımcı olabilir.

Örnek: Bir proje ekibi, belirli bir tedarikçi tarafından temin edilen malzemelerin gecikmesi riskini yönetmek için alternatif tedarikçilerle anlaşma yapabilir.

Adım 3.3 Riskin Transferi:

Riskin transferi, riskin gerçekleşmesi durumunda oluşabilecek zararı karşılayacak çözümler bularak (örneğin sigorta yaptırmak) riskin başkalarına aktarılmasıdır. Riskin diğer bir partiye taşınması çok karşılaşılan bir yöntemdir ancak bu riski azaltmamaktadır. Ancak bu aktarım hemen hemen her

zaman ek bir ödemeyle sonuçlanmaktadır. Sabit ödemeli anlaşmalarda riskin proje sahibinden bir yükleniciye transfer edilmesi tipik bir örnektir.

Riskin transferi stratejisi, projede ortaya çıkabilecek risklerin etkilerini ve maliyetlerini başka bir tarafta devretmeyi amaçlar. Riskin transferi stratejisinin bazı önemli yönleri ve örnekleri:

Sigorta Yaptırma:

Sigorta, en yaygın risk transferi yöntemlerinden biridir. Projede olası risklere karşı sigorta poliçeleri satın alınarak, risk gerçekleştiğinde sigorta şirketi zararı karşılar.

Örnek: Bir inşaat projesi sahibi, proje sırasında olası inşaat kazalarına veya doğal afetlere karşı sigorta poliçeleri satın alarak bu riskleri transfer edebilir.

Taşeron Sözleşmeleri:

Proje sahibi, belirli işleri taşeronlara veya alt yüklenicilere devrederek riskleri transfer edebilir. Taşeronlar, belirli bir işi yerine getirmekle yükümlüdür ve bu işi zamanında ve bütçe dahilinde tamamlamak zorundadır.

Örnek: Bir yazılım projesi sahibi, belirli bir yazılım bileşenini bir yazılım firmasına taşeron olarak vererek bu bileşenin geliştirme riskini transfer edebilir.

Sözleşme Şartları:

Proje sözleşmeleri, belirli risklerin kimin sorumluluğunda olduğunu ve hangi şartlar altında riskin transfer edilebileceğini belirler. Sözleşme şartları, risklerin transferini düzenler.

Örnek: İnşaat sözleşmesi, belirli bir projenin süresini aşarsa, ek maliyetlerin taşeron tarafından karşılanmasını şart koşabilir.

Alt Yüklenici Anlaşmaları:

Proje sahibi, belirli iş paketlerini veya bileşenleri alt yüklenicilere devredebilir. Alt yükleniciler, bu işleri yerine getirme ve ilgili riskleri üstlenme konusunda uzmanlaşmıştır.

Örnek: Bir inşaat projesi sahibi, beton döküm işini uzman bir alt yükleniciye devredebilir, böylece beton dökümü ile ilgili riski transfer etmiş olur.

Adım 3.4 Riskin Paylaştırılması:

Riskin paylaştırılması, riskin belli oranlarla farklı partilere bölüştürülmesidir. Bunun bir örneği olarak Airbus A340 verilebilir. Araştırma ve geliştirme riskleri İngiltere ve Fransa'yı da içeren Avrupa ülkeleri arasında bölüştürülmüştür.

Riskin paylaştırılması stratejisinin bazı önemli yönleri ve örnekleri:

Ortak Girişimler:

Birden fazla şirket veya kuruluş arasında gerçekleştirilen ortak girişimler, projedeki risklerin paylaştırılmasına bir örnektir. Her bir ortak, projenin başarısızlık durumunda olası kayıpları taşır.

Örnek: Bir enerji santrali inşaat projesi, enerji şirketi ve inşaat şirketi arasında bir ortak girişim olarak gerçekleştirilebilir. Her iki şirket, projenin başarısızlık riskini paylaşır.

Konsorsiyum Anlaşmaları:

Büyük projelerde, farklı firmalar arasında konsorsiyumlar oluşturularak riskler paylaştırılabilir. Konsorsiyum üyeleri, projenin farklı yönlerini veya bileşenlerini üstlenirler.

Örnek: Bir uluslararası inşaat projesi için, inşaat, mühendislik ve lojistik alanlarında uzmanlaşmış farklı firmalar bir araya gelerek konsorsiyum oluşturabilirler.

Devlet ve Özel Sektör İşbirlikleri:

Projelerde devlet ve özel sektör işbirliği sıkça görülür. Bu tür işbirlikleri, projenin maliyetlerini ve risklerini paylaştırır.

Örnek: Bir otoyol inşaatı projesi, devlet ve özel sektör işbirliğiyle gerçekleştirilebilir. Devlet, projenin finansmanını sağlarken özel sektör işletme ve işletme riskini üstlenebilir.

Alt Yüklenici ve Tedarikçilerle Anlaşmalar:

Projelerde farklı iş paketleri veya bileşenleri alt yüklenicilere veya tedarikçilere verilerek riskler paylaştırılabilir. Bu alt yükleniciler, belirli işleri tamamlamak ve bu işlere ilişkin riskleri üstlenmekle sorumludur.

Örnek: Bir inşaat projesinde, çelik çerçeve montajı işi bir alt yükleniciye verilerek çelik yapı riski transfer edilebilir.

Riskin paylaştırılması stratejisi, projedeki riskleri farklı taraflar arasında adil bir şekilde bölüştürmeyi amaçlar. Bu strateji, her bir tarafın kabul edilebilir bir risk yükünü taşımasına olanak tanır. Ancak, bu tür anlaşmaların ayrıntıları ve risklerin dağıtımı özenle ele alınmalıdır, aksi takdirde anlaşmazlıklar ve sorunlar ortaya çıkabilir.

Adım 3.5 Riskin Kabul Edilmesi:

Bu aşamada risk ihtimalini veya etkisini etkilemek için herhangi bir önlem alınmamaktadır. Bazı durumlarda meydana gelen bir olayın riskini kabul etmek bilinçli bir yaklaşımdır. Bazı risk olayları transfer edilemeyecek ya da azaltılamayacak kadar büyük olabilir (örneğin deprem, su baskını vs). Proje sahibi bu tip riskleri kabul eder çünkü bu tip olayların meydana gelme olasılıkları oldukça zayıftır. Diğer durumlarda ek bütçe yapılırken belirlenmiş olan risklerin etkisi kolayca bu bütçeden karşılanabilecektir.

Proje başlamadan önce riske tepki geliştirme işine daha büyük çaba harcanması projede karşılaşılabilecek sürprizlerin olumsuz etkilerinin minimize edilme olasılığını artırır.

Riskin kabul edilmesi stratejisinin bazı önemli yönleri ve örnekleri:

Doğal Afetler:

Bazı projeler coğrafi olarak doğal afetlere maruz kalabilirler (örneğin, depremler veya sel). Bu tür afetler büyük ölçüde önceden tahmin edilemez ve projede kabul edilen bir risk olabilir.

Örnek: Bir deniz kenarı tatil köyü projesi sahibi, olası tsunami riskini kabul ederek, projesi için uygun güvenlik önlemleri alabilir.

Piyasa Dalgalanmaları:

Piyasa dalgalanmaları, özellikle finansal projelerde önemli bir risk olabilir. Fiyat dalgalanmaları veya faiz oranlarının değişiklikleri gibi faktörler projeyi etkileyebilir.

Örnek: Bir enerji üretim projesi sahibi, enerji fiyatlarının dalgalanmalarını kabul ederek, bu dalgalanmaları göz önüne alarak finansman stratejisi oluşturabilir.

Teknolojik Belirsizlik:

Yeni teknolojilerin veya ürünlerin geliştirilmesi projelerde teknolojik belirsizliğe yol açabilir. Bu tür belirsizlikler projede kabul edilen bir risk olabilir.

Örnek: Bir araştırma ve geliştirme projesi sahibi, yeni bir ürün geliştirme sürecinde teknik sorunlarla karşılaşma riskini kabul edebilir.

İşgücü Sorunları:

İşgücü eksikliği veya nitelikli personel bulma zorluğu, projelerde risk oluşturabilir. Bu tür sorunlar projede kabul edilen bir risk olarak ele alınabilir.

Örnek: Bir inşaat projesi sahibi, nitelikli inşaat işçileri bulma konusundaki zorlukları kabul ederek, projenin zaman çizelgesini ayarlayabilir.

Riskin kabul edilmesi stratejisi, projede karşılaşılabilecek olası sorunların farkında olmayı ve bu sorunların etkilerini minimize etmek için uygun önlemleri almamayı içerir. Bu strateji, bazı risklerin transfer edilemeyeceği veya azaltılamayacağı durumlarda kullanışlıdır. Ancak, risklerin dikkatle analiz edilmesi ve projenin kabul edilebilir riskleri sınırlar içinde tutması önemlidir.

1.2.2.5. Kaynak Kullanımı Yönetimi

Kaynak kullanımı yönetimi, proje bütçesinin harcandığı süreçtir. Gerçekleşen harcamanın planlanan harcamaya uygunluğunun belirlenmesi amacıyla kaynak kullanımı ölçülmeli ve raporlanmalıdır. Harcamalara ek olarak üretkenlik izlenmelidir. Bu kapsamda kazanılan değer analizi (EVA) adı verilen teknik kullanılmaktadır.

EVA, proje sırasında aşağıdaki ölçütlerin düzenli aralıklarla karşılaştırılmasını içerir:

- Bugüne kadar gerçekleşen bütçe,

- Bugüne kadar gerçekleşen fiili harcama,

- Tamamlanacak tahmin,

- Tamamlanma süresi tahmini.

EVA, yaygın biçimde kullanılan bir performans ölçüm tekniğidir. Projenin kapsamı, maliyeti ve zaman çizelgesi ölçümlerini bütünleştirerek proje yönetim ekibinin proje performansını ve ilerlemesini değerlendirmesine ve ölçmesine yardımcı olan grafiksel bir araçtır.

Projenin yürütülmesi sırasında sapmalar ile karşılaşması hâlinde proje yöneticisinin cevabını araştırması gereken beş temel soru bulunmaktadır:

- Sapmaya neden olan problem nedir?

- Sapmanın zaman, maliyet ve performans üzerindeki etkisi nedir?

- Sapmanın, eğer var ise diğer çabalar üzerindeki etkisi nedir?

- Planlanan veya uygulanan düzeltici hareket nedir?

- Düzeltici hareket ya da hareketlerin beklenen sonuçları nedir?

Proje ekibi bu soruların cevaplarına göre gerekli düzenlemeleri yaparak projenin başarı şansını arttırabilir. Bilgi sistemleri denetçisinin proje yöneticisi tarafından bu hususların değerlendirilip değerlendirilmediğini incelemesi beklenir.

Kazanılan Değer Analizi (EVA) yönteminin avantajları ve dezavantajları şu şekildedir:

Avantajları:

Performans Değerlendirmesi: EVA, proje performansını objektif bir şekilde değerlendirmek için kullanılır. Bu sayede proje yöneticileri, projenin bütçe, zaman ve kapsam hedeflerine ne kadar yaklaştığını anlayabilirler.

Entegre Ölçüm: EVA, maliyet, zaman ve kapsam gibi farklı ölçümleri bir araya getirerek projenin genel performansını gösterir. Bu, proje yöneticilerinin farklı alanlardaki sapmaları daha iyi anlamalarını sağlar.

Erken Uyarı Sistemi: EVA, projede olası sapmaları erken teşhis etme imkanı sunar. Bu, proje yöneticilerinin düzeltici önlemleri zamanında almasına yardımcı olur.

Karar Verme Araçları: EVA, proje yöneticilerine gelecekteki kararlarını desteklemek için kullanabilecekleri veriler sağlar. Örneğin, projenin son tarihine ulaşması için ek kaynakların gerekip gerekmediği konusunda bilgi sunabilir.

Dezavantajları:

Karmaşıklık: EVA'nın hesaplanması ve yorumlanması karmaşık olabilir. Özellikle büyük ve karmaşık projelerde bu işlem zaman alabilir.

Veri İhtiyacı: EVA'nın kullanılabilmesi için doğru ve güncel verilere ihtiyaç vardır. Eğer veriler eksik veya yanlışsa, analiz sonuçları güvenilir olmayabilir.

Öğrenme Eşiği: EVA'yı etkili bir şekilde kullanabilmek için projeye dahil olan ekiplerin bu yöntemi öğrenmeleri gerekebilir. Bu da zaman ve kaynak gerektirebilir.

1.2.2.6. Proje Faydalarının İzlenmesi

Her proje fayda yaratmak amacıyla gerçekleştirilir. Projeler için proje yaşam döngülerinden öte yeni oluşturulan sistem için tüm iş fayda ve toplamdaki iş maliyetlerini değerlendiren uzun planlamalara yönelik fayda geliştirecek şekilde planlanmış bir yaklaşıma ihtiyaç bulunmaktadır. Faydalar nadiren planlarda öngörüldüğü şekillerde ortaya çıkar. Proje faydalarının gerçekleştirilmesinin temel unsurları aşağıdaki gibidir:

Faydaların Yönetiminin Açıklanması: Proje ekibi ve paydaşları için projenin neden yapıldığı ve hangi faydaları sağlamayı amaçladığı açıkça belirtilmelidir. Proje amacı ve hedefleri net bir şekilde iletilmelidir.

Örnek: Bir şirket, yeni bir müşteri ilişkileri yönetimi (CRM) sistemini uygulamaya koymayı planlıyor. Bu projeyle, müşteri memnuniyetini artırmak ve satışları artırmak amaçlanıyor. Projeyi başlatmadan önce, projenin neden yapıldığı ve hangi faydaları sağlamayı amaçladığı tüm paydaşlara açıkça iletilir.

Bir Ölçüm ve Hedef Atama: Her bir faydanın nasıl ölçüleceği ve hangi hedeflere ulaşılması gerektiği belirlenmelidir. Bu, projenin ilerlemesini izlemek için temel bir çerçeve sağlar.

Örnek: Bir belediye, trafik sıkışıklığını azaltmayı hedefleyen bir ulaşım projesi yürütüyor. Bu projenin başarısını ölçmek için trafik sıkışıklığının yoğunluğu ve seyahat sürelerindeki değişiklikler gibi belirli ölçütler belirlenir.

Bir İzleme/Ölçüm Rejiminin Oluşturulması: Proje süresince faydaların izlenmesi ve ölçülmesi için bir rejim veya süreç oluşturulmalıdır. Bu, faydaların zaman içinde nasıl geliştiğini anlamak için gereklidir.

Örnek: Bir sağlık kuruluşu, yeni bir hasta kayıt sistemi uyguluyor. Proje süresince, sistemin etkinliğini izlemek için hasta randevularının süresi ve veri giriş hataları gibi ölçümler düzenli olarak kaydedilir.

Varsayım Belgelemesi: Projede kullanılan tüm varsayımların belgelenmesi önemlidir. Varsayımların doğru olduğundan emin olunmalı ve değişiklikler izlenmelidir.

Örnek: Bir inşaat projesi için belirli malzemelerin belirli bir tedarikçiden temin edileceği varsayımı yapılır ve bu varsayım yazılı olarak belgelendirilir.

Gerçekleştirme İçin Anahtar Sorumlulukların Oluşturulması: Faydaların gerçekleştirilmesi için belirli sorumluluklar atanmalıdır. Bu, kimin ne yapması gerektiğini netleştirir.

Örnek: Bir yazılım geliştirme projesinde, yazılım testleri ve kalite kontrolü için belirli ekipler ve kişiler sorumlu tutulur.

İşte Öngörülen Faydaların Doğrulanması: Proje sonunda, öngörülen faydaların gerçekten elde edilip edilmediği değerlendirilmelidir. Bu, projenin başarısını ölçmek için önemlidir.

Örnek: Bir eğitim kurumu, eğitim materyallerinin dijitalleştirilmesi projesini tamamlar. Proje sonrasında, öğrenci başarılarının artırılıp artırılmadığını değerlendirmek için sınav sonuçları ve öğrenci geri bildirimleri analiz edilir.

- Gerçekleştirilmesi Gereken Fayda Planlanması: Proje sonuçlarına ulaşıldığında faydaların nasıl gerçekleştirileceği planlanmalıdır. Bu, faydaların sadece projenin sonunda değil, aynı zamanda sonrasında da elde edilmesini sağlar.

Örnek: Bir enerji şirketi, enerji verimliliğini artırmayı amaçlayan bir proje yürütüyor. Proje sonrasında enerji maliyetlerini düşürmek için enerji yönetimi ekipleri oluşturur ve bu ekiplerin görevleri belirlenir.

Projede faydanın gerçekleşme aşamaları;

- Anlama: Bu aşamada, projenin neden yapılması gerektiği ve projenin ne tür faydalar sağlayacağı net bir şekilde anlaşılmalıdır. Proje hedefleri ve amaçları belirlenirken, hangi faydaların elde edilmesi gerektiği açıkça tanımlanır. Proje başlamadan önce, tüm paydaşlar arasında projenin amaçları ve beklentileri hakkında bir anlayış oluşturulur.

Örnek: Bir yazılım şirketi, bir müşteri ilişki yönetimi (CRM) yazılımı geliştirmeyi planlıyor. Anlama aşamasında, proje yöneticileri ve yazılım geliştirme ekibi, müşterinin iş ihtiyaçlarını ve hedeflerini anlamak için müşteri ile görüşmeler yapar. Müşteri, müşteri verilerini daha iyi yönetme, satış süreçlerini iyileştirme ve müşteri memnuniyetini artırma hedefleri belirtir.

- Planlama: Faydaların planlama aşaması, proje planının oluşturulması ve projenin nasıl yönetileceğinin belirlenmesini içerir. Bu aşamada, hangi kaynakların kullanılacağı, proje takvimi, bütçe ve risk yönetimi planları gibi önemli detaylar belirlenir. Aynı zamanda fayda ölçümünün nasıl yapılacağı ve faydaların hangi ölçütlerle değerlendirileceği planlanır.

Örnek: Proje planlaması sırasında, yazılım geliştirme ekibi, CRM yazılımının gereksinimlerini ve özelliklerini belirler. Ayrıca projenin zaman çizelgesi, bütçesi ve kaynakları planlanır. Proje planı, yazılım geliştirme aşamalarını ve test süreçlerini ayrıntılı olarak açıklar.

- Gerçekleştirme: Proje gerçekleştirme aşamasında, proje planına göre çalışmalar başlatılır ve projenin uygulanması başlar. Bu aşamada, fayda sağlama süreci başlar. Yani, projenin gerçekleştirilmesi sırasında, belirlenen faydaların elde edilmesi için gerekli adımlar atılır. Proje ekibi, plana sadık kalmak ve hedeflenen faydaları elde etmek için çalışır.

Örnek: Yazılım geliştirme aşamasında, yazılım mühendisleri müşterinin belirlediği gereksinimlere göre CRM yazılımını kodlarlar. İşlevselliği eklerken, kullanıcı dostu bir arayüz tasarımına da dikkat ederler. Yazılım testleri yapılır ve hatalar düzeltilir. Sonunda, CRM yazılımı müşteriye teslim edilir ve kullanıma başlanır.

- Raporlama: Proje süresince ve sonrasında, fayda sağlanma süreci düzenli olarak izlenir ve raporlanır. Bu, proje yöneticilerinin ve paydaşların fayda sağlama ilerlemesini ve performansını takip etmelerini sağlar. Faydaların ne kadarının elde edildiği, plana göre ilerleme ve herhangi bir sapma raporlar aracılığıyla değerlendirilir. Raporlama aynı zamanda düzeltici önlemlerin alınmasına ve fayda sağlama sürecinin iyileştirilmesine olanak tanır.

Örnek: CRM yazılımının kullanımı başladıktan sonra, yazılımın performansı düzenli olarak izlenir. Müşterinin satış verileri, müşteri memnuniyeti ve satış süreçleri takip edilir. Raporlar, yazılımın müşterinin iş hedeflerine ne kadar katkı sağladığını değerlendirmek için kullanılır. Eğer belirlenen hedeflere ulaşılmıyorsa, yazılım geliştirme ekibi düzeltici önlemler alır ve yazılımı günceller.

Proje faydalarının gerçekleştirilmesi maliyet, kalite, geliştirme/teslim süresi, güvenilirlik ve bağımlılık gibi önemli faktörler arasında uzlaşmadır. Stratejistler kapsamlı bir çalışma yürütür ve elemeyi geçen/kazanan faktörleri değerlendirir. Ardından bu faktörlerin sistemleri tamamlamak ve bakımını yapmak için mevcut hizmetlerin güçlü, zayıf yönleri ile karşılaştırır. Birçok işletme bilgi sistemlerindeki değişiklikleri desteklemek için yapılandırılmış proje yönetimi ilkelerini kullanmaktadır. Bilgi sistemleri denetçisi işletmenin geliştirmeye ilişkin projeler için değeri veya yatırım getiri hedeflerini tutarlı bir şekilde yerine getirememesi, yazılım geliştirme yaşam döngüsündeki (YGYD-Software Develeopment Life Cycle-SDLC) ve ilgili proje yönetimi pratiklerindeki zayıflığını gösterebilir.

1.2.2.7. Maliyet Yönetimi

Proje büyüklüğüne bakılmaksızın maliyetin kontrol altında tutulması ve bütçenin aşılmaması, proje yöneticisinin her zaman titizlikle üzerinde durması gereken bir husustur. Maliyet izleme ve kontrol işlemleri, projenin tamamlanması amacına doğru gidilen yolda, yöneticinin projeyi plana uygun bir biçimde tamamlamasına yardımcı olmalıdır. Etkin bir maliyet yönetim sisteminde aşağıdaki unsurların yer alması gerekmektedir:

- Maliyet tahmini,

- Maliyet muhasebesi,

- Proje nakit akışı,

- Proje paydaşları nakit akışı,

- Direkt iş gücü maliyeti,

- Cezalar, kâr paylaşımı vb.

Maliyetin etkin kontrolünün sağlanması için izleme ve kontrol süreçlerinin aşağıdaki unsurları içermesi gerekir:

- Projenin tamamlanması için yapılacak tüm işlerin planlamasının iyi bir biçimde yapılmış olması,

- İş gücü ve maliyetlere ilişkin güçlü ve güvenilir tahminlerin yapılmış olması,

- Görevler ve kapsam arasındaki ilişkilerin iyi belirlenmiş olması,

- Disiplin altına alınmış bir bütçe ve harcama yetkisinin bulunması,

- Fiziksel ilerleme ve maliyet harcamaları için muhasebe işlemlerinin zamanında yürütülmesi,

- Kalan işin tamamlanması için gereken zaman ve maliyet için periyodik olarak tahminlerin yürütülmesi,

- Projenin doğasına uygun aralıklarla gerçekleşen iş ve maliyetler ile planlanan iş ve maliyetler arasında karşılaştırmalar yapılması.

Maliyetlerin kontrolü için oluşturulan bir izleme ve kontrol sistemi sayesinde proje yöneticisi, planlanan işler ile gerçekleşen işleri karşılaştırabilme imkânını elde etmektedir. Bu karşılaştırmalar sayesinde;

- Amaçların başarılı biçimde performans standardı hâline gelip gelmediği,

- Gerçek değerler ve planlanan değerler arasındaki farkın ortaya çıkmasına olanak verecek bir bütçenin hazırlanıp hazırlanmadığı,

- Performans standartlarının proje faaliyetlerinin güvenilir birer temsilcisi olup olmadığı.

Plan dâhilinde yer alan faaliyetin tamamlanması veya ek maliyet talebi ile ortaya çıkan maliyet değerlerinin derlenmesi maliyet kontrol sisteminin en önemli işlevidir. Gerçekleşen maliyet (PMBOK'a göre AC) ve tamamlanan işin bütçelenen maliyeti (PMBOK'a göre EV) her bir faaliyet için ayrıntılı olarak ele alınır ve planlanan değer (PMBOK'a göre PV). Maliyet Veri Derleme ve Raporlama Akış Diyagramı'na göre raporlanır (Ş 7.2; S.183).

Maliyet kontrolünün içermesi gereken ögeler aşağıda sıralanmıştır:

- Onaylanan maliyette değişikliklere yol açan faktörlerin ele alınması,

- Tüm değişiklik taleplerinin zamanında işleme konulması,

- Gerçekleşen değişikliklerin, gerçekleştikleri sırada yönetilmesi,

- Maliyet harcamalarının gerek dönem gerekse proje bazında onaylanan bütçeyi aşmamasının sağlanması,

- Onaylanan maliyete göre farklılıkların belirlenmesi ve bunların anlaşılması için maliyet performansının izlenmesi,

- Çalışma performansının, harcanan fonlarla karşılaştırılarak izlenmesi,

- Onaylanmayan değişikliklerin, raporlanan maliyet ya da kaynak kullanımına dâhil edilmesinin önlenmesi,

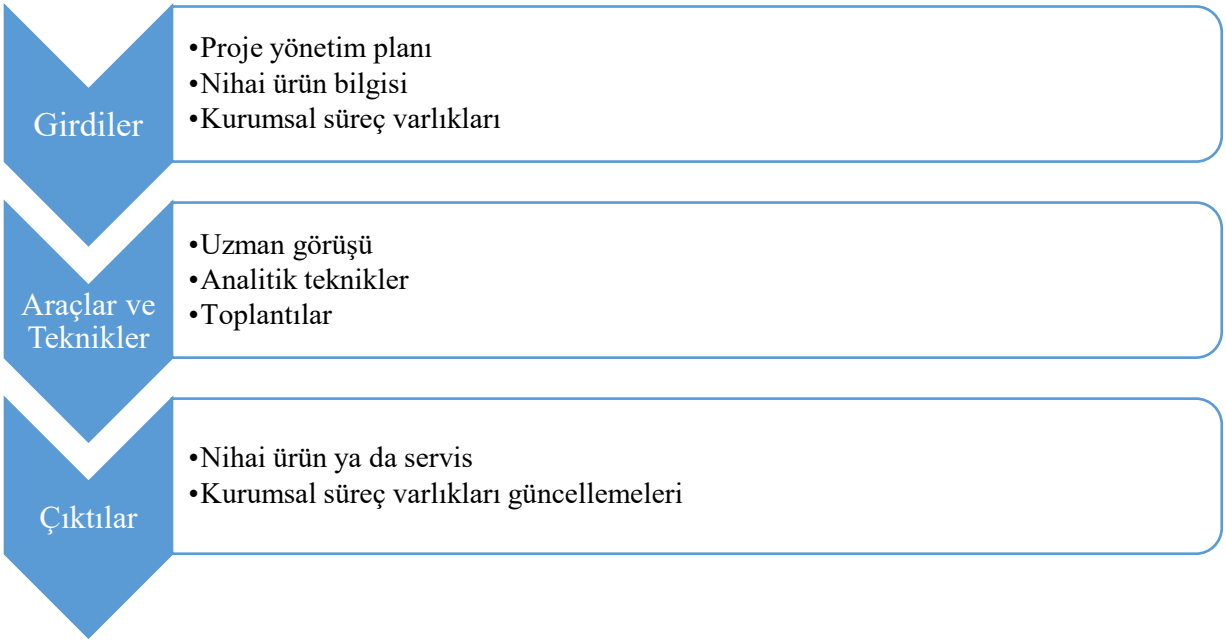
- Paydaşların, onaylanan tüm değişiklikler ve bunlarla ilgili maliyetler konusunda bilgilendirilmesi,

- Beklenen maliyet aşımlarını, kabul edilir sınırlar içinde tutmaya gayret gösterilmesi.

1.2.3. Projenin Kapatılması

Projenin kapatılması, proje akışı içerisinde yer alan son süreçtir. Projenin kapatılması aşamasına verilecek önem, proje yöneticisinin ve paydaşların daha sonraki projelerinde daha başarılı olmaları olasılığını arttıracaktır. Proje yöneticisi, projenin yaşam döngüsünün başlaması ile birlikte projenin kapatılması için gerekli belgeleri de hazırlamaya başlamış olmaktadır. Projenin gerçekleştirilmesi sırasında ortaya çıkan problemlerin nasıl çözümlendiğine ilişkin raporlar, kapsamda meydana gelen değişimlere ilişkin notlar, paydaş katkıları, kalite raporları, bütçe değişikliklerine ilişkin bilgi ve belgeler, proje süresince proje takımının edindiği deneyimler/kazanımlar/dersler bir bütün olarak projenin kapatılması aşamasının öğelerini oluşturmaktadır. Projenin kapatılması aşamasının başarılı olması için proje yöneticisinin bu aşamayı da bir süreç olarak yönetmesinde büyük fayda sağlayacaktır.

Projenin kapatılması sürecinin girdileri, araç ve teknikleri ve çıktıları aşağıda belirtilmiştir:



Proje Kapatma Girdi, Araç ve Teknikler ve Çıktılar Şeması

Proje bir noktada kapatılmalı ve bu aşamada kullanıcılara ve/veya sistem destek personeline yeni veya değiştirilmiş bir sistem teslim edilmelidir. Bu noktada, tespit edilen sorunların atanması gerekir. Proje sponsorunun, üretilen sistemi kabul edilebilir olması veya teslime hazır olduğundan memnun olmasının sağlanması önemlidir. Bu kapsamda, dikkate alınması gereken temel sorular aşağıdaki gibidir:

- Proje yöneticisi, nihai proje kapanış bildirimini ne zaman yayımlayacaktır?

- Nihai proje bildirimini kim yayımlayacaktır?

- Proje yöneticisi, proje ekibinin yeni projelere geçişine ya da düzenli atanmış görevlerine bırakılmasına nasıl yardımcı olacaktır?

- Proje yöneticisi açık kalan eylemler, riskler ve sorunlar için ne yapacaktır? Bu eylemleri kimler üstlenecek ve bunlar nasıl finanse edilecektir?

Bu aşamada ilgili belgeleme ve görevlerin teslimi gerçekleşecek, proje sahibinin sonucu onaylaması istenecektir. Onay için performans, test ve kontrol sonuçları dikkate alınacaktır. Proje kapanışında projenin başarısı veya projeden çıkarılan dersler proje ekibi ve ilgili paydaşlar tarafından belirlenecektir.

Projenin kapatılabilmesi için proje yaşam çevrimi boyunca yapılan tüm anlaşmaların kapatılması gerekmektedir. Aslında tedariklerin kapatılması alt süreci projenin başlaması ile başlayarak, projenin kapatılması ile sona ermektedir. Tedarik kapatma alt sürecine, müşteriden ürün ile ilgili onayın alınması da dâhildir. Tedarik kapatılırken müşterinin de onayı alınmaktadır. Müşteri onayı alınırken

genellikle iki durum ortaya çıkmaktadır. İlk olarak müşteri ve proje ekibi arasında toplantı düzenlenmeli ve toplantıda müşterinin ürünü kabul etmesi önemlidir. Müşteri ürünün beklentilerini karşıladığını ve belgede belirtilen tarihte ürünü teslim aldığını ifade eden metni imzalanır. Eğer proje herhangi bir neden ile ürün ortaya çıkmadan sonlandırılıyor ise yine proje yöneticisi ve müşteri arasında projenin sonlandırıldığına dair bir belge imzalanır. Bu durumda proje sonlandırma nedenleri belgeye eklenir. Son aşamada ortaya çıkabilecek problemlerin çözümlerinin maliyetinin yüksek olma ihtimali göz önüne alınmalı ve bütçe planlamasında son aşamada ortaya çıkabilecek öngörülemeyen problemler için bir harcama kalemi bulundurulmalıdır. Projenin kapatılması ile beraber tüm sözleşmeler, muhasebe kayıtları arşive aktarılır.

Öğrenilmiş dersler belgesi oluşturulurken, proje yöneticisi temel olarak kendisi ve ekibinin karşılaştığı problemleri belirlemekte ve bu problemlerden gelecek projelerde nasıl kaçınabileceğini kurgularsa çok daha faydalı bir belge ortaya çıkmaktadır. Proje yöneticisi karşılaştığı her problem için izleyen kesimde sıralanan soruları sorarak gelecekte benzer sorunlar altında kalmayacak veya benzer sorunlar ile karşılaşıldığında ne yapacağını bilecektir. Proje kapsamında tanımlanmış bir problem için sorulabilecek sorular:

- Problem ve etkisi nedir?

Bu problemden dolayı projede ne tür sorunların ortaya çıktığının öğrenilmesi, problemin tüm açılardan ele alınması gerekmektedir.

- Problemin ortaya çıkma nedeni ne olabilir?

Problemin nedenini bulmak için algılanan veya bilinen etkinin ne olduğu araştırılmalıdır. Problemin nedeni araştırılırken harcanacak zaman ve maliyetlerin problemin çözümü ile elde edilerek faydaya değip değmeyeceği araştırılmaktadır.

- Problem neden daha önce tespit edilemedi?

Projenin izleme ve kontrol süreci doğru kurgulanmamış olabilir. Bu nedenle izleme ve kontrol sistemlerinin doğru çalıştığından emin olunmalıdır. Ayrıca performans raporlarının doğru yazıldığı ve ilgili bireylere zamanında ulaştırılıp ulaştırılmadığı araştırılmalıdır.

- Problem kişisel performansa bağlı olabilir mi?

Söz konusu problem sadece bir proje çalışanının performans düşüklüğünden kaynaklanmış olabilir.

- Gelecekte bu problem bertaraf edilebilir mi?

Bazen söz konusu problemin tekrar ortaya çıkmasını engellemek imkânsız olabilir. Bu durumda gelecekte aynı problem ile karşılaşmamak için projede yapılması mümkün tüm değişikliklerin neler olduğu tespit edilmelidir.

- Bertaraf edilemez bir problem ise erken tespit mümkün müdür?

Bu aşamada proje ekibi ile beraber izleme, kontrol, raporlama ve kalite kontrol süreçleri yardımıyla problemin ortaya çıkışının mümkün olan en erken sürede anlaşılabilmesi için alınması gereken tedbirler gözden geçirilmelidir.

Proje süresince ortaya çıkan problemlerin çözüme kavuşturulması için sıralanan bu soruların cevapları yardımıyla öğrenilmiş dersler ortaya çıkmaya başlamaktadır. Projenin kapatılması aşamasında da bu dersler bir belge içerisinde bir araya getirilerek gelecekte aynı problemler ile karşılaşıldığında çözüme giden yol kısaltılmış olmaktadır. Öğrenilmiş dersler belgesi hazırlanırken projenin büyüklük ve tipine göre kategoriler oluşturulabilmektedir. Bu kategoriler içerisinde yer alan her bir öğrenilmiş ders bilgisi tek tek sıralanmaktadır.

Nihai proje raporu, proje yaşam çevrimi sürecince ortaya çıkan tüm gelişmeleri içeren bir belgedir. Rapor yardımıyla projenin üzerinden çok uzun bir süre geçse dahi proje süresince yaşananların açık ve anlaşılır bir şekilde hatırlanabilmektedir. Aynı şekilde benzer projelere başlayacak olan proje yöneticileri için de nihai proje raporu yol gösterici bir belge olmaktadır. İyi hazırlanmış bir nihai proje

raporunun projeye ilişkin birçok bilgiyi içermesi beklenmekle birlikte temel olarak rapor içerisinde yer alması gereken ana başlıklar izleyen biçimde sıralanabilmektedir.

- Projenin Genel Başarısı: Paydaşlardan, müşteriden ve proje çalışanlarından elde edilen geri bildirimler göz önüne alındığında proje ne kadar başarıya ulaşmıştır? Başarı için kullanılan ölçütler nelerdir ve proje kapsamında bu proje ölçütlerine ne kadar ulaşılmıştır?

- Projenin Genel Organizasyonu/Düzeni: Projenin tamamlanması ile beraber proje yöneticisi, projenin genel organizasyonunu nasıl gerçekleştirdiğini gözden geçirme şansını yakalamaktadır. Proje yöneticisi gerçekten iyi bir organizasyon gerçekleştirilip gerçekleştirilmediğini araştırmaktadır ve bir sonraki projede nelere dikkat edilmesi gerektiğini sıralamaktadır.

- Projenin Güçlü ve Zayıf Yanları: Proje yöneticisi, bir güçlü ve zayıf yönler, fırsatlar ve tehditler (SWOT) analizi veya proje yaşam çevrim süreci boyunca tutulan not ve raporlar yardımıyla projenin güçlü ve zayıf yanlarını ortaya çıkararak raporuna ekler. İleride raporun hazırlandığı proje konusunda yapılacak yeni çalışmalar için varsa öneriler ayrıntılandırılarak listelenmektedir.

- Proje Ekibine İlişkin Tavsiyeler: Projenin başlatılmasından kapatılmasına kadar geçen süre zarfında proje ekibine ilişkin bilgi alışverişleri olması kaçınılmazdır. Proje ekibi içerisinde bireyselliği ön planda tutanlar ile takım çalışmasına yatkın olanlar belirlenerek çeşitli tavsiye kararları sıralanabilmektedir.

- Sonuç Veren Teknikler: Projede sonuç veren tekniklere ilişkin özet bir liste hazırlanır. Bu liste hazırlama işine projenin başlaması ile hayat verilir. Sonuçların veya çıktıların elde edilebilmesi için ortaya konan çabaların neler olduğuna dair bilgiler bu liste içerisinde yer almalıdır.

Nihai proje raporunun hazırlanmasında paydaş ve müşteri görüşlerinin de yer almasına özen gösterilmelidir. Projenin kapatılması sürecinde proje nihai raporu oluşturulması amacıyla izlenen içindekiler yapısı oluşturulabilir. Projenin türüne ve büyüklüğüne göre rapor daha da ayrıntılı hâle getirilebilir. Bu raporda aşağıdaki hususlara yer verilir.

- Proje adı ve tarih: Projenin açık ismi, varsa proje numarası ve raporun hazırlanma tarihi belirtilir.

- Projenin kapatılmasına ilişkin betimleyici istatistikler: Gerçekleşen maliyet değeri, plana göre projenin tamamlanma tarihi, ekibin performansı ve kalite başarım miktarı belirtilmektedir.

- Varsa ilk faydalar: Proje sayesinde elde edinilen faydaların birçoğu ilk anda ortaya çıkmayabilir. Eğer proje kapatılması sırasında kazanılmamış faydalar olduğu düşünülüyor ise proje yöneticisinin proje kapatıldıktan ve bu faydalar ortaya çıktıktan sonra bir rapor hazırlanması beklenir. Bu rapora proje tespit raporu adı verilir.

- Öğrenilmiş dersler: Hazırlanmış öğrenilmiş dersler belgesi nihai rapor içerisine eklenir.

- Proje yöneticisinin projeye ilişkin görüş ve önerileri: Proje sürecinde nelerin yolunda gittiği, nelerin yolunda gitmediği ve kazanılmış dersler kapsamına girmeyen ama projeye ilişkin olarak değerlendirilmesinde fayda olan konular ele alınır.

Proje yöneticisi, projenin başlangıcından itibaren iyi bir evraklama sistemine ihtiyaç duyar. Bu evraklama sistemi bilgisayar ortamında olabileceği gibi dosya içerisinde basılı belgelerin ve faturaların bir kütüphanede tasnif edilmesi biçiminde de gerçekleşebilir. Önemli olan proje için üretilen tüm belge ve faturaların kronolojik ve ilgi alanlarına göre indekslenip bir arada tutulmalarıdır. Evraklama sistemi sayesinde projenin yaşam çevrimi süresince projenin nasıl geliştiği ve sona doğru ilerlendiği görülecektir. Eğer proje çeşitli sebeplerden dolayı sonlandırıldı ise sonlandırma nedenine kaynak olabilecek proje tabanlı gelişmelere ilişkin izler de eski evrakların incelenmesi ile tespit edilebilir. Düzgün ve kolay anlaşılır bir arşiv sistemi oluşturulması projenin başarısına katkı sağlayacaktır.

Bir değişiklik yönetim sistemi izleme bilgileri aşağıdaki hususları içermelidir:

- Programcıların oturum açma kapama geçmişi, program silme tarihçesi, görevler ayrılığı (SoD) ve kalite güvence faaliyetlerinin geçmişi ile atanan programcı, iş emri tarihi, yapılan değişiklikler ve kapanış tarihi gibi detayların yer aldığı iş emri faaliyetlerinin geçmişi,

- Bütünlüğün sağlanması amacıyla üretim kütüphanesi güvenliğinin yeterliliğinin belirlenmesi amacıyla mevcut kontrollerin incelenmesi,

- Kontrol hedeflerinin sağlandığından emin olunması için sistem bakım süreçlerinin değerlendirilmesi, test sonuçları ve denetim kanıtlarının dökümante edilmesi,

- Sistem bakım standartları ve prosedürlerin kilit personel ile gözden geçirilmesi,

- Uygulama sonrası gözden geçirmelere katılınması.

Proje yönetiminde bilgi sistemleri denetçisinin rolü proje süresince veya proje kapanışı tamamlandıktan sonra gerçekleştirilebilir. Bu kapsamda;

- Kontrol gerektiren alanların belirlenmesi amacıyla sistemin ana bileşenleri, hedefleri, kullanıcı gereksinimlerinin belirlenmesi için anahtar sistem geliştirme ve proje paydaşları ile tanışılması,

- Sistemin risklerinin ve zayıf yönlerinin belirlenmesi ve sıralanması için sistem geliştirme ve proje paydaşları ile uygun kontrol yöntemlerinin belirlenmesi,

- Sistem riskinin azaltılması için önleyici kontrollerin belirlenmesi,

- Mevcut kontrollerin değerlendirilmesi ve işlerliğinin proje paydaşları ile incelenmesi,

Kontrollerin uygulandığı, gereksinimlerin karşılandığından ve sistem geliştirme/satın alma metodolojisinin takip edildiğinden emin olunabilmesi amacıyla tüm sürecin, belge ve teslimatların incelenmesi, tüm değişiklik, düzenleme ve işleme işlemlerine ilişkin kontrollerin tanımlandığından emin olunması.

Örnek Sorular

Soru 1: Etkin maliyet yönetimi aşağıdaki unsurlardan hangilerini içermez?

A) Maliyet tahmini

B) Proje nakit akışı

C) Direkt iş gücü maliyeti

D) Cezalar, kâr paylaşımı vb.

E) Proje analiz süreçleri

Cevap: E

Soru 2: Bir bilgi sistemleri denetçisi, aşağıdakilerden hangisinin proje yöneticisi tarafından izlenip kontrol altına alındığına bakmalıdır?

A) Proje planı ile karşılaştırmalı olarak hâli hazırda yürütülmekte olan faaliyet ya da faaliyetlerin durumunun tespiti

B) O ana kadar tamamlanan iş miktarı ve kalitesi

C) Proje planı ile karşılaştırmalı olarak maliyetlerin ve harcamaların durumu

D) Proje ile ilgili tüm paydaşların (proje çalışanları, işverenler, müşteriler vb.) projeye karşı olan genel tavır ve hareketleri

E) Hepsi

Cevap: E

Soru 3: Bir bilgi sistemleri denetçisi Proje ekibinde kalite kontrol için nelerin yapıldığına bakabilir?

A) İstatistiksel kalite kontrol teknikleri

B) Hatalı ürünlerin müşteriye ulaşmasına engel olacak denetlemeler

C) Kalitede anormalliklere ve dalgalanmalara yol açan nedenlerin tespiti

D) Kalitenin istenen seviyede olması için gerekli kontrol sınırlarının belirlenmesi

E) Hepsi

Cevap: E

Soru 4: Bir bilgi sistemleri denetçisi kazanılan değer analizi (EVA) tekniği ile proje sırasında aşağıdaki ölçütlerden hangisinin düzenli aralıklarla karşılaştırılmasını içerdiğinden emin olmalıdır?

A) Bugüne kadar gerçekleşen bütçe

B) Bugüne kadar gerçekleşen fiili harcama

C) Tamamlanacak tahmin

D) Tamamlanma süresi tahmini

E) Hepsi

Cevap: E

Soru 5: Riskin yönetilmesi sorumluluğunun başka bir tarafa aktarılmasını sağlayan riske yanıt stratejisi aşağıdakilerden hangisidir?

A) Riski aktarma

B) Riskten kaçınma

C) Riski kabullenme

D) Riski azaltma

E) Duruma göre davranma

Cevap: A

Soru 6: Bir bilgi sistemleri denetçisi projenin SMART metoduna uygunluğunu incelerken aşağıdaki bileşenlerden hangisini göz önünde bulundurmaz?

A) Ölçülebilir

B) Belirli bir amaca hizmet eden

C) Süreli

D) Gerçekçi

E) Belirsiz

Cevap: E

Soru 7: Projelerde zaman, maliyet ve kapsam yönetimi önemli üç önemli kısıttır. Bir projede kapsam sabit kalmak şartı ile proje maliyeti düşürülmek istenirse, proje süresi genellikle nasıl değişim gösterir?

A) Proje süresi kısalır.

B) Proje süresi değişmez.

C) Proje süresi uzar.

D) Proje maliyeti hiçbir zaman düşürülemez.

E) Kapsam hiçbir zaman sabit kalmaz.

Cevap: C

Soru 8: Aşağıdakilerden hangisi BS denetçisinin iş kırılım yapısı ve ilgili çalışma paketlerine ilişkin göz önünde bulundurması gereken temel konulardan değildir?

A) En üst iş kırılım seviyesi nihai teslimatı veya projeyi temsil etmektedir.

B) Bir iş kırılımının tüm öğelerinin aynı seviyede tanımlanması gerekmemektedir.

C) Münferit çalışma paketleri alt teslimatları üretmek için gereken görevlerin işini, süresini ve maliyetlerini tanımlamaktadır.

D) Münferit çalışma paketleri ortalama 10 günü aşmamalıdır.

E) Münferit çalışma paketleri birbirine benzer, iş kırılımı genelinde çoğaltılabilir.

Cevap: E

Soru 9: Aşağıdakilerden hangisi kritik yolu kısaltacak önlemler arasında sayılamaz?

A) Kritik etkinlikler üzerine yoğunlaşılması

B) Seri işlerin paralel olarak yeniden planlanması (Hızlı izleme)

C) Kritik etkinliklerin sürelerinin uzatılması

D) Çok kaynak kullanan etkinliklerin kısaltılması

E) Maliyeti yüksek işlerin kısaltılması

Cevap: C

Soru 10: Bir bilgi sistemleri denetçisi aşağıdakilerden hangisinin proje yönetim süreçleri arasında olmasını beklemez?

A) Başlatma

B) Planlama

C) Yürütme

D) Kontrol Etme

E) Bekleme

Cevap: E

Soru 11: Aşağıdakilerden hangisi tipik bir projede olması gereken özelliklerden biri değildir?

A) Her bir projenin sonucunda benzersiz bir ürün, hizmet veya sonuç ortaya çıkar.

B) Projeler geçici bir süre içinde gerçekleştirildikleri için her projenin bir başlangıcı ve bitişi vardır.

C) Projelerin sürekli kontrol edilmesine gerek yoktur.

D) Projelerin tanımlanmış bir bütçeleri ve hedefleri vardır.

E) Projeleri oluşturan faaliyetler benzersizdirler.

Cevap: C

Soru 12: Aşağıdakilerden hangisi projede yer alan grup/bireylerin rol ve sorumlulukları arasında yer alır?

A) Proje sponsoru

B) Kalite güvence

C) Proje Yöneticisi

D) Üst Düzey Yönetim

E) Hepsi

Cevap: D

Soru 13: Aşağıdakilerden hangisi bir bilgi sistemleri denetçisinin proje yönetimi denetimleri sırasında proje yönetim ekibinin kalite kontrol için kullanmasını bekleyeceği beceri ve teknikler arasında yer alır?

A) İstatistiksel kalite kontrol teknikleri

B) Hatalı ürünlerin müşteriye ulaşmasına engel olacak denetlemeler

C) Kalitede anormalliklere ve dalgalanmalara yol açan nedenlerin tespiti

D) Kalitenin istenen seviyede olması için gerekli kontrol sınırlarının belirlenmesi

E) Hepsi

Cevap E

Soru 14: Aşağıdakilerden hangisi proje portföy yönetiminin hedefleridir?

A) Bireysel olmayan proje portföy sonuçlarının iyileştirilmesi,

B) Projelerde önceliklendirme ve zaman planlamasının yapılması

C) İç ve Dış proje kaynak koordinasyonlarının sağlanması

D) Proje süresince bilgi akışının sağlanması

E) Hepsi

Cevap E

Soru 15: Aşağıdakilerden hangisi fizibilite çalışmasının alt maddesidir?

A) Proje Kapsamı

B) Mevcut analiz

C) Gereksinimler

D) Yaklaşım

E) Hepsi

Cevap: E

2. SİSTEM GELİŞTİRME YAŞAM DÖNGÜSÜ (SYSTEM DEVELOPMENT LIFE CYCLE, SDLC)

Sistem geliştirme yaşam döngüsü (SDLC), bir sistem geliştirme projesinin tüm aşamalarını içeren bir yöntemdir. SDLC, sistem geliştirme sürecinin her bir safhasını ayrıntılı olarak planlamak, yürütmek ve kontrol etmek için bir çerçeve sağlar. Böylece, geliştirme sürecindeki hataları azaltılarak projenin başarısı artırılır.

Son yıllarda, SDLC sürecinin hızlandırılması ve verimliliğinin artırılması için yeni teknolojiler kullanılmaktadır. Özellikle yapay zeka ve makine öğrenmesi, süreçlerin daha hızlı ve doğru bir şekilde yürütülmesini sağlar.

Bilgi sistemlerinde SDCL, bilgi sistemlerinin planlanması, kontrollü bir şekilde geliştirilmesi, test edilmesi ve uygulanmasına ilişkin süreçtir. Sistemlerin zamanla karmaşıklaşması ve iş ihtiyaçlarının değişmesi sonucunda sistem geliştirme metodolojileri yaygın bir şekilde kullanılmaktadır. Çevik geliştirme (Agile) ve Geliştirme ve Operasyonlar (DevOps) gibi metodolojiler, SDLC'yi daha çevik hale getirmek için kullanılır. Bu metodolojiler, yazılımın sürekli olarak geliştirilmesine ve hızlı bir şekilde dağıtılmasına olanak tanır.

SDLC, bir yazılım veya bilgi sistemi oluşturmak, geliştirmek veya iyileştirmek için izlenen aşamaları ve süreçleri tanımlayan bir metodolojidir. SDLC, bir projenin başlangıcından sonuna kadar olan tüm adımları kapsar ve her bir aşamanın özel hedefleri, görevleri ve çıktıları vardır. Bu, bir projenin düzenli bir şekilde ilerlemesini ve sonuçlarını yönetilmesini sağlar.

Günümüzde, SDLC'nin her aşamasında yapay zeka destekli araçlar kullanılmaktadır. Bu araçlar, proje yönetimi, kodlama, test etme ve bakım aşamalarında yazılımın kalitesini artırabilir.

SDLC, genellikle aşağıdaki temel aşamalardan oluşur:

Planlama: Bu aşama, projenin hedeflerinin ve kapsamının belirlendiği yerdir. Proje planı oluşturulur, bütçe tahminleri yapılır, kaynaklar atanır ve projenin genel stratejisi belirlenir. Planlama aşamasında, bulut tabanlı çözümler ve yapay zeka destekli yazılımlar sayesinde kaynaklar hızlıca tahsis edilir ve proje stratejileri optimize edilir.

Analiz: Bu aşamada, mevcut iş süreçleri ile sistemler analiz edilir. Projenin gereksinimlerini toplamak ve anlamak için yapılır. Müşteriler ve paydaşlarla iletişim kurularak sistem gereksinimleri belirlenir. Yapay zeka, analiz aşamasında büyük veri analizi ile daha doğru sistem gereksinimleri belirlemeye yardımcı olabilir. Ayrıca, makine öğrenmesi, sistemin kullanımını ve gereksinimlerini öğrenerek geliştirme sürecinde önemli bir rehber olabilir.

Tasarım: Analiz aşamasında belirlenen gereksinimlere dayanarak sistem tasarımı yapılır. Bu, sistem mimarisi, veri tabanı yapısı, kullanıcı arayüzü tasarımı ve diğer teknik ayrıntıları içerir. Bu aşama, bulut tabanlı altyapı çözümleri ile desteklenerek, esnek ve ölçeklenebilir sistem tasarımlarının yapılmasına imkan tanır.

Geliştirme: Bu aşamada, tasarlanan sistemin gerçek kodlaması ve yazılım geliştirmesi yapılır. Programlama dilleri ve teknolojiler kullanılarak uygulama oluşturulur. Geliştirme aşamasında, DevOps ve Agile metodolojileriyle sürekli entegrasyon (CI) ve sürekli dağıtım (CD) uygulanarak yazılımın hızlı ve sürekli geliştirilmesi sağlanır.

Test Etme: Geliştirilen yazılımın kalitesini ve işlevselliğini doğrulamak için kapsamlı testler yapılır. Bu, hata ayıklama, performans testleri ve güvenlik testleri gibi çeşitli testleri içerir. Test etme aşamasında, otomasyon araçları ve yapay zeka destekli test çözümleri kullanılarak test süreçleri hızlandırılabilir ve daha doğru sonuçlar elde edilebilir.

Uygulama: Bu aşama, yazılımın canlı ortama uygulandığı ve kullanılmaya başladığı yerdir. Kullanıcılar ve yöneticiler için eğitim verilir ve sistem devreye alınır. Bulut tabanlı uygulama çözümleri, bu aşamanın daha hızlı ve sorunsuz bir şekilde gerçekleştirilmesine olanak sağlar. Yazılım uygulaması, canlı ortamda anında güncellenebilir ve bakım yapılabilir.

Bakım ve Destek: Yazılımın kullanımı sırasında ortaya çıkan sorunları gidermek, güncellemeleri uygulamak ve performansı optimize etmek için düzenli bakım ve destek sağlanır.

SDLC'nin bakım ve destek aşamasında, yazılımın bulut tabanlı sistemlere entegrasyonu, sürekli bakım ve güncellemelerin kolayca yapılmasına olanak tanır.

Her aşama, projenin ilerlemesini ve sonuçlarını değerlendirmek için özel kontrol noktaları içerir ve proje ekibi tarafından tamamlanmalıdır. SDLC, projelerin düzenli bir şekilde yönetilmesini, kaynakların etkili bir şekilde kullanılmasını ve son kullanıcı ihtiyaçlarına uygun çözümlerin üretilmesini sağlayan bir çerçeve sağlar. Ayrıca, herhangi bir projenin özel gereksinimlerine ve sektöre uyacak şekilde özelleştirilebilir.

Modern yazılım geliştirme araçları ve metodolojileri, SDLC'yi daha verimli hale getirmek için sürekli olarak gelişmektedir. Yapay zeka, DevOps, bulut çözümleri ve otomasyon gibi yenilikçi araçlar, yazılım geliştirme süreçlerinde önemli iyileştirmeler sağlamaktadır.

2.1. Sistem Geliştirme Süreçleri

Sistem geliştirme süreçlerinin her aşaması bir sonraki aşama için temel teşkil etmektedir. Bu sayede iş uygulamalarına ilişkin yürütülen geliştirme aşamaları daha etkin bir yönetim kontrolü sağlamaktadır. İş ve sistemlerin entegrasyonu yönetişimin en önemli unsurlarındandır. İş uygulaması geliştirme süreci, üretime alma ve yaygınlaştırma, bakım faaliyetleri ve ömrünü tamamlamış sistemlerin üretim ortamından kaldırılması aşamalarını içermektedir.

Bir iş uygulaması geliştirme projesi, aşağıdaki durumlardan biri veya birkaçı sonucunda başlatılır:

Yeni/Mevcut İş Süreciyle İlgili Oluşan Yeni Fırsat: İşletme, yeni bir pazarlama stratejisi, müşteri hizmetleri yaklaşımı veya verimlilik artırma fırsatı gördüğünde, bu fırsatı değerlendirmek için yeni bir iş uygulaması geliştirme projesi başlatılabilir. Aşağıda konunun daha iyi anlaşılması için konu ile alakalı örnekler verilmiştir:

Örnek 1: Bir e-ticaret şirketi, müşteri verilerini daha iyi analiz etmeye olanak tanıyan yeni bir veri madenciliği aracının piyasaya sürülmesini fırsat olarak görüp, bu aracı kullanarak kişiselleştirilmiş ürün önerileri sunan bir proje başlatır.

Örnek 2: Bir sağlık hizmetleri sağlayıcısı, hastaların tıbbi kayıtlarını daha etkili bir şekilde yönetmeyi amaçlayan bir elektronik tıp kaydı (EMR) sistemini uygulamak için bir proje başlatır.

Mevcut Teknolojiye İlişkin Bir Sorun: İşletme, mevcut sistemlerin veya yazılımların işletme gereksinimlerini karşılayamadığını veya sürekli olarak sorunlar yaşandığını fark ederse, bu sorunları çözmek için bir geliştirme projesi başlatılabilir. Aşağıda konunun daha iyi anlaşılması için konu ile alakalı örnekler verilmiştir:

Örnek 1: Bir finansal kuruluş, mevcut müşteri bilgi yönetim sistemi tarafından yavaş ve veri güncellemelerinde sık sık hatalı sonuçlar verildiği için yeni bir müşteri bilgi yönetim sistemi geliştirmek için bir proje başlatır.

Örnek 2: Bir üretim şirketi, eski bir üretim hattının sık sık arızalanması ve üretkenlik kaybına neden olması nedeniyle daha yeni ve verimli bir üretim hattı kurmak için bir proje başlatır.

Bu tür projelerde, yapay zeka (AI - Yapay Zeka) ve makine öğrenmesi (ML - Makine Öğrenmesi), sistemlerin daha verimli ve hatasız çalışması için çözüm sunar. Yapay zeka, veri analizi ve hata tespiti konusunda büyük bir rol oynar. Ayrıca, bulut tabanlı çözümler kullanılarak sistemler daha esnek hale getirilebilir, veriler anında erişilebilir ve depolanabilir. Bu modern teknolojiler, sistem entegrasyonu, verimlilik artışı ve hataların azaltılması için etkili araçlar sunar.

Özellikle, API tabanlı entegrasyonlar (API - Uygulama Programlama Arayüzü), farklı sistemlerin birbiriyle uyumlu çalışmasını sağlar ve işletmelerin farklı yazılım sistemlerini entegre etmelerini kolaylaştırır. Bu süreç, iş uygulamaları geliştirme ve entegre etme aşamalarını daha verimli hale getirir.

Örnek: Bir eğitim kurumu, eski ve güncel olmayan öğrenci yönetim sistemi yazılımını kullanıyor. Bu yazılım, öğrenci kayıtlarını yönetmek ve notları saklamak için kullanılıyor ancak sürekli olarak güncellenmiyor ve modern eğitim ihtiyaçlarına cevap veremiyor. Özellikle öğrenci bilgilerinin

güvenliği ve erişilebilirliği konusunda sorunlar yaşıyorlar. Ayrıca, bu eski yazılım, öğrenci bilgilerini sık sık kaybediyor ve bu da öğrenci memnuniyetsizliğine ve karmaşık manuel iş süreçlerine yol açıyor. Bu nedenle, eğitim kurumu, güncel ve daha güvenilir bir öğrenci yönetim sistemi uygulama projesini başlatarak bu sorunları çözmeyi amaçlıyor.

İş Uygulamalarının İş Ortakları/Endüstri Standardı Sistemler ve İlgili Ara Yüzlerle Uyumlu Hale Getirilmesi: İşletme, iş ortaklarıyla veya endüstri standartlarıyla uyumlu olmayan iş uygulamalarına sahipse veya yeni bir entegrasyon gerekliliği ortaya çıkarsa, bu uyumluluğu sağlamak için yeni bir proje başlatılabilir. Aşağıda konunun daha iyi anlaşılması için konu ile alakalı örnekler verilmiştir:

Örnek 1: Bir e-ticaret platformu, ödeme işlemleri için mevcut bir ödeme ağının gereksinimlerine uyum sağlayamadığı için daha yaygın olarak kullanılan bir ödeme yöntemi ile entegre olma projesi başlatır.

Örnek 2: Bir kamu kurumu, farklı departmanlar arasında bilgi paylaşımını kolaylaştırmak için farklı yazılım sistemlerini birleştirmek ve uyumlu hale getirmek için bir entegrasyon projesi başlatır.

Mevcut İş Süreciyle İlgili Gelişen Sorun: İşletme, mevcut iş süreçlerinin aksaklıklarını veya sorunlarını belirlediğinde, bu sorunları gidermek ve süreçleri optimize etmek için bir geliştirme projesi başlatabilir. Aşağıda konunun daha iyi anlaşılması için konu ile alakalı örnekler verilmiştir:

Örnek 1: Bir lojistik şirketi, envanter yönetimi süreçlerinin karmaşıklığı ve hatalı envanter sayımları nedeniyle daha etkili bir envanter yönetim sistemi kurmak için bir proje başlatır.

Örnek 2: Bir otomobil üreticisi, üretim hattındaki sık sık gecikmeler ve üretim hataları nedeniyle üretim süreçlerini optimize etmek için bir iyileştirme projesi başlatır.

İşletmenin teknolojiden faydalanmasını sağlayacak yeni bir fırsat: Yeni bir teknoloji veya platformun işletmenin verimliliğini artırabileceği veya yeni iş fırsatları yaratabileceği durumlarda, bu fırsatları değerlendirmek için bir proje başlatılabilir.

Örnek 1: Bir telekomünikasyon şirketi, yeni 5G teknolojisi ile daha hızlı ve güvenilir hizmetler sunma fırsatını değerlendirmek için bir ağ güncelleme projesi başlatır.

Örnek 2: Bir medya şirketi, artan sanal gerçeklik (VR) teknolojisinin eğlence içeriklerini zenginleştirmek için bir VR içerik üretimi projesi başlatır.

Örnek 3: Bir e-ticaret şirketi, yapay zeka tabanlı öneri sistemlerini kullanarak müşterilere daha kişiselleştirilmiş ürün önerileri sunma fırsatını görüyor. Bu sistem, müşterilerin geçmiş alışveriş alışkanlıklarını ve tarayıcı geçmişlerini analiz ederek, onlara ilgilerine en uygun ürünleri öneriyor. Bu, müşteri memnuniyetini artırmanın yanı sıra satışları ve geliri de artırma potansiyeline sahiptir.

Örnek 4: Bir otomotiv üreticisi, otonom sürüş teknolojileri konusunda büyük bir fırsat görüyor. Gelecekte, otonom araçlar sürücüsüz olarak yolculuk yapabilecek ve bu da sürüş güvenliğini artırmanın yanı sıra trafik verimliliğini artıracak. Bu teknolojiyi benimsemek, rekabet avantajı sağlayabilir ve tüketici taleplerini karşılayabilir.

İş uygulamaları geliştirilmesi iki farklı kategoride değerlendirilmektedir:

1) Organizasyon Odaklı (Merkezli): Organizasyon odaklı uygulamaların temel amacı, bilinmesi gerekenler ilkesine yönelik kullanıcıların ve destek fonksiyonlarına ilişkin bilgilerin konsolide edilmesi, depolanıp arşivlenmesi ve paylaşılmasıdır. Çoğunlukla bu tür projelerde geliştirme için SDLC veya detaylı yazılım mühendisliği yaklaşımları da kullanılmaktadır. Bu tür uygulamalara aşağıdaki örnekler verilebilir:

Muhasebe ve Finans Yönetimi Yazılımı: İşletmeler, finansal verileri takip etmek, gelir-gider tablolarını oluşturmak ve bütçe yönetimini kolaylaştırmak için organizasyon odaklı bir muhasebe yazılımı kullanabilirler. Bu tür yazılımlar, işletme içinde finansal bilgileri merkezileştirir ve yöneticilere finansal kararlar için gereken bilgileri sunar.

İnsan Kaynakları Bilgi Sistemi (HRIS): Büyük organizasyonlar, işe alım, eğitim, personel performansı ve maaş yönetimi gibi insan kaynakları süreçlerini merkezileştirmek ve izlemek için HRIS kullanabilirler. HRIS, personel verilerini tutar, raporlar oluşturur ve işe alım süreçlerini kolaylaştırır.

Envanter Yönetim Sistemi: Bir perakende işletmesi, envanterin alınması, depolanması, takip edilmesi ve yönetilmesi için organizasyon odaklı bir envanter yönetim yazılımı kullanabilir. Bu tür yazılımlar, stok düzeylerini optimize etmeye ve tedarik zincirini yönetmeye yardımcı olabilir.

Proje Yönetimi Yazılımı: Büyük inşaat projeleri veya yazılım geliştirme projeleri gibi karmaşık projeleri yönetmek için organizasyon odaklı bir proje yönetimi yazılımı kullanılır. Bu yazılımlar, görevlerin atanması, ilerlemenin izlenmesi ve kaynakların tahsis edilmesi gibi projelerin yönetimini kolaylaştırır.

2) Son Kullanıcı Odaklı (Merkezli): Son kullanıcı odaklı uygulamaların amacı ise, performansın optimize edilmesini sağlamak için farklı veri görünümleri sağlamaktır. Bu amaç, karar destek sistemlerini (DSS), coğrafi bilgi sistemlerini ve ilgili teknikleri vb. içermektedir. Bu tür uygulamaların çoğu alternatif yaklaşımlar kullanılarak geliştirilmektedir. Bu tür uygulamalara aşağıdaki örnekler verilebilir:

Karar Destek Sistemleri (DSS): DSS, yöneticilere karmaşık veri analizi yapma ve stratejik kararlar almaları için yardımcı olan son kullanıcı odaklı bir uygulama kategorisine örnektir. Bu sistemler, işletme verilerini görselleştirebilir, analiz edebilir ve yöneticilere çeşitli senaryoları değerlendirmeleri için araçlar sunar.

Coğrafi Bilgi Sistemleri (GIS): GIS, coğrafi verileri analiz etmek, haritalamak ve görselleştirmek için kullanılan bir teknolojidir. Özellikle konum tabanlı işlerde, son kullanıcılar için haritalama ve coğrafi analiz yapma olanağı sağlar. Örneğin, bir şehir yönetimi, trafik yönetimi veya yerel hizmetler için GIS kullanabilir.

Müşteri İlişkileri Yönetimi (CRM) Yazılımı: Bir şirket, müşteri ilişkilerini yönetmek ve müşteri verilerini analiz etmek için son kullanıcı odaklı bir CRM yazılımı kullanabilir. Bu yazılım, müşteri etkileşimlerini izler, satış fırsatlarını takip eder ve müşteri hizmeti süreçlerini optimize eder.

Tıbbi Görüntüleme Yazılımları: Sağlık sektöründe, doktorlar ve sağlık uzmanları için son kullanıcı odaklı tıbbi görüntüleme yazılımları kullanılır. Bu yazılımlar, MR, CT taramaları ve röntgenler gibi tıbbi görüntüleri işlemek ve analiz etmek için kullanılır.

SDLC modelleri aşağıdaki 3 başlıkta verilebilir.

1) Geleneksel Şelale (Waterfall) Modeli

Zaman içinde yazılım projelerinin büyüyerek bir proje yönetim metodolojisi gerektirmesi, kritik sistemlerde hatalara tolerans gösterilemeyecek olması, teknolojinin gelişerek bilgisayar kapasitelerinin hızla artması ve kaliteli yazılım üretilememesi gibi nedenlerden dolayı 1960'lı yıllarda yaşanan yazılım krizi sonucu 1968 yılında düzenlenen Yazılım Mühendisliği Konferansı'nda üzerinde durulan temel konular yazılım mühendisliği kavramının tanımlanması ve yazılım geliştirme süreçlerinde uygulanması gereken standartların tespit edilmesi olmuştur. Daha sonra 1970 yılında Dr. Winston Royce tarafından yayınlanan bir bildiride Şelale Modeli tanımlanmıştır. Her ne kadar önerilen metodoloji evrimsel ve yinelemeli süreçler ile geliştirme süresince müşterinin katılımı gibi yaklaşık 30 yıl sonra "*çevik*" olarak adlandırılacak uygulamalara vurgu yapsa da bu model zaman içinde bugün bilinen şeklini almış ve yazılım geliştirme metodolojilerinin temeli haline gelmiştir.

Şelale modeli proje yönetim süreci; sıralı ve doğrusal bir proje yönetim süreci olup, BT projeleri ve yazılım mühendisliği için SDLC'nin en popüler versiyonudur. Kullanımı karmaşık olmayan projeler için daha uygundur. Geleneksel bir yöntemdir. Süreçler tıpkı bir şelale gibi yukarıdan aşağıya doğrusal olarak işlemektedir. Analiz, tasarım, yazılım, test, yayın gibi fazlardan oluşur. Bir faz tamamlanıp yenisine geçildiğinde, bir önceki faza geri dönülmez. Söz konusu aşamanın yeniden gözden geçirilmesi ancak birinci aşamadan başlanmasıdır. Bir aşamanın çıktısı bir sonraki aşamanın girdisidir. Proje sonunda elde edilen ürün müşteriye teslim edilmektedir.

Söz konusu model potansiyel hataların yalnızca nihai kabul testi öncesinde düzeltilmesini sağlayan bir yaşam döngüsü doğrulama yaklaşımını kapsamaktadır. Bu durum projenin gereksinimlerinin net ve iyi tanımlanmış olması durumunda son derece efektiftir. Ancak iletişimin tek yönlü olması nedeniyle gereksinimlerin belirlenmesi veya analiz aşamasında müşteri ihtiyacının net olarak anlaşılmaması hatalı geliştirmelere neden olabilecektir. Bu sebeple müşteri tamamlanan yazılım sistemini tüm artı ve eksileriyle kabullenmek ve kullanmak zorundadır. Geleneksel yaklaşım, gereksinimlerin ve tasarımın tamamlanmasına yardımcı olmak için ekran prototiplerinin gerekli olduğu web uygulamalarında kullanışlıdır.

Proje yönetim modelleri genelde şelale modeli ile başlar. Yazılım mühendisliğinde önemli bir modeldir. Şelale modeli çok daha statiktir. Geleneksel yaklaşımın en önemli avantajı, iş, tasarım ve programlama gereksinimleri için bir şablon sağlamasıdır.

Bu yaklaşımın avantajları aşağıdaki gibidir:

- Projenin başlangıcında gereksinimlerin açık, anlaşılır ve tam olarak tanımlanması, daha doğru zaman ve maliyet tahmini yapılmasını sağlayarak projedeki belirsizlikleri azaltır.

- Proje yönetim ve ekip rollerinin belli olması; kimin ne zaman ne yapması gerektiği katı bir şekilde tanımlandığı ve takip edildiği için kendi başına iş yapma yeterliliği ve tecrübesi olmayan proje elemanları, proje yöneticileri ve performansı iniş çıkışlı proje ekipleri için idealdir.

- Bir safhaya başlanmadan önce bir önceki safha tamamlandığı için proje ilerleyişinin takibi kolaydır.

- Disiplinli ve istikrarlı bir model olduğu için proje elemanlarına projenin iyi bir şekilde yönetildiği ve her şeyin kontrol altında olduğu hissiyatını verir.

- Kullanımı ve yönetimi kolaydır.

- Projede müşteri kaynak ihtiyacı daha azdır.

- Müşterinin projeye ne zaman ve ne kadar katkı yapacağı bellidir ve proje boyunca zaman ayırması beklenmez. Fonksiyonel organizasyonlarda birden fazla projede görev alan proje elemanlarının işlerinin üst üste binmesini engeller. Örneğin analiz çalışması tamamlandıktan sonra başka bir projede görev almaya devam eden bir elemanın geri dönüşü gerekmez.

- Çok kullanılan bir yöntem olduğu için genelde teoride iyi bilinir.

- İşletmenin klasik yöntemlere aşina olması nedeniyle genel olarak proje teknikleri ile ilgili olarak eğitim ihtiyacının agile tekniklere göre daha azdır.

- Projenin her aşamasında ayrıntılı belgelerin üretilmesini gerektirir. Bu, projenin geçmişini ve gereksinimlerini daha iyi belgelemenizi sağlar.

- Her aşama için tanımlanmış net roller ve sorumluluklar vardır. Bu, proje ekibinin görevlerini ve sorumluluklarını net bir şekilde anlamalarına yardımcı olur.

- Her aşama tamamlandığında, bir sonraki aşama için hazır olunur. Bu, projenin aşamalarının sırayla tamamlanmasını kolaylaştırır.

Bu yaklaşımın dezavantajları aşağıdaki gibidir:

- Müşterinin ürünü projenin sonlarına doğru, geliştirme aşamasından sonra gördüğünde çok sayıda değişiklik isteğinin çıkabilmektedir.

- Bir safhanın tamamlanmadan sonraki safhanın başlamaması nedeniyle, tüm proje elemanlarını ilgilendirmeyen gecikmeler dahi bütün proje elemanları için zaman israfına neden olabilmektedir.

- Herhangi bir safhada tespit edilmemiş bir olumsuzluk sonraki bütün safhaları etkilemektedir.

- Değişiklik, geliştirme veya düzeltme amaçlı geriye dönüşler şelalede akıntıya karşı yüzmeye benzetilir, zor ve maliyetlidir. Bu yüzden değişikliklere cevap veremez ve süreç içerisinde gelen istekler çoğunlukla kabul edilmez. Bu nedenle müşteri ihtiyaçlarına cevap verilmediği ölçüde projede kapsamın kayması maliyet artışı getireceği ve süreyi uzatacağı için başarısızlık olasılığı yükselebilmektedir.

- Gereksinimlerin proje başında tam ve anlaşılır bir şekilde tanımlanmasının beklenmesi çok gerçekçi olmayabilir ve müşterilerin ihtiyaçlarını tam ve doğru ifade edememesi proje sonunda eksik veya yanlış bir ürün elde edilmesine neden olabilir.

- Geliştiricilerin gereksinimleri müşteri veya kullanıcılarla yüz yüze görüşerek anlamaları yerine analiz ve tasarım safhalarında hazırlanan dokümanlardan okumaları, gereksinimlerin anlaşılmasını zorlaştırır, geliştiricilerin girdi yapmasına imkân vermez.

- Gereksinim tutarsızlıkları, eksik sistem bileşenleri ve beklenmeyen geliştirme ihtiyaçları, ancak tasarım veya kodlama aşamasında belirlenebilir.

- Sistem performansı tüm kodlama tamamlanana kadar ölçülemez ve tüm sistem testine kadar hatalar tespit edilemez.

- Yoğun ve güncel dökümantasyon zaman ve emek gerektiren bir faaliyettir ve genel olarak geliştiriciler tarafından gereksiz bir yük olarak görülür.

- Ürün, kullanıcılar tarafından yeterince sahiplenilmediği için kullanımında direnç gösterilebilmektedir.

- Yazılım geliştirme faaliyetleri yaratıcılık gerektiren faaliyetlerdir ve şelale modelinin yüksek disiplinli yaklaşımı yaratıcılığı olumsuz etkileyebilmektedir.

- Değişikliklere ve esnekliğe karşı dirençlidir. Proje başladıktan sonra gereksinimlerde veya tasarımda değişiklikler yapmak zordur ve maliyetlidir.

- Müşteriler ürünü genellikle projenin sonunda görürler. Bu nedenle, müşterinin ihtiyaçlarını tam olarak anlamak ve karşılamak zor olabilir.

- Değişiklikler projenin ilerleyişini geriye döndürebilir ve maliyeti artırabilir. Bu nedenle, gereksinimlerin baştan net ve doğru bir şekilde belirlenmesi önemlidir.

- Büyük ve karmaşık projeler için uygun olmayabilir, çünkü her aşama tamamlanana kadar sonuçlar görülemeyebilir ve değişiklik yapmak zor olabilir.

- Yüksek yönetim giderleri küçük projeler ve küçük takımlar için gereksiz maliyetlerdir.

2) V-Şekil Modeli

Adımlar V şeklini oluşturduğu için V model ya da V şekil model denilmiştir. Bu model, doğrulama (verification) ve geçerleme (validation) modeli anlamına gelir. Şelale modelinde olduğu gibi katı bir disipline sahip olan bu modelde de bir sonraki aşamanın başlangıcı, önceki aşamanın bitişi ile mümkündür. Ancak şelale modelinden farklı olarak proje basamakları lineer olarak aşağıya doğru gitmek yerine, her modelin eşleniği olarak bir test evresini barındırmak sureti ile V şeklini çizen bir şemaya sahiptir. Şeklin sol tarafı doğrulama kısmını oluştururken, sağ tarafı geçerleme kısmını oluşturur. Model, ilk önce şelale modelinde olduğu planlama evresi ile başlar ve şelale modelindeki adımları kodlama aşamasına kadar aynı şekilde takip eder. Ancak kodlama aşamasından sonra yukarıya doğru yönelen lineer gelişim evresi, çoğunlukla kontrol ve test süreçlerini kapsayan ve test süreçleri sonucunda uygulamaya geçen bir süreci temsil eder. Her test süreci, aslında gelişim evresinde karşılık bulduğu adımın bir testi niteliğindedir.

Bu model, aslında şelale modeline benzerdir. Planlama, ihtiyaç belirleme, üst seviye ve alt seviye tasarımlar vardır. Ancak bazı farkları bulunmaktadır. Üst seviye de daha genel bakış açısına göre tasarım yapılır. Üst seviye tasarımların daha sonra daha detaylı alt seviye tasarımlarının yapıldığı görülür. Daha alt seviyelerinin açıldığı, girdi-çıktıları ve beklentilerinin yazıldığı söylenebilir. Sonrasında kodlamaya geçilir. Buraya kadar şelale modeli ile benzerlikler söz konusudur. Aşağı doğru akan bir proje yönetimi söz konusudur. Buradan sonra yukarı doğru çıkılmaya başlanır. Birim testleri yapılır. Her üretilen modülün testi yapılır. Daha sonra bu modüllerin/alt seviye tasarım ürünlerinin birbiri ile çalışma durumu, entegrasyon testleri yapılır ve kabul testleri aşamasında müşterinin kabul edip etmemesi ile ilgili müşteriye gidilir. Müşteri gereksinimler kapsamında uygulamayı test eder. Sonra bakım başlar. Bu da proje geliştirme modeli olarak V şekil modeli diğer modellere göre daha güvenilir

hale getirir. Ancak bu güvenilirlik, proje oluşumunda yapılan araştırmaların derinliğine ve kalitesine bağlı kalır.

Önceden öngörülemeyen risklerin sonraki aşamalarda probleme dönüşümü, bu proje modeli için en temel risk unsurunu oluşturur. Bu risk unsurunun nedeni olan kısıtlı esneklik yapısı, Çevik model ile karşılaştırıldığında bu model için en temel zayıf noktadır. Bu nedenle iyi planlanmış büyük sistemler veya planlanması çok karmaşık olmayan küçük sistemler için tercih edilebilen bir yöntem olmasına rağmen, her zaman öngörülemeyen risk unsurunu barındırır. Bunun yanında Çevik modeli ile benzer olan bir yanı mevcuttur. Bazı durumlarda Çevik modelin testleri, tasarımın oluşturulma aşamasında belirlenebilir. Bu gibi çevik model projeleri, testlerin oluşturulma zamanları ile V şekil modele benzerlik göstermektedirler (Radack, 2009).

Bilgi sistemleri denetçisinin bakış açısından, V şekil modeli aşağıdaki avantajları sağlar:

- Bilgi sistemleri denetçisinin etkisi, iş uygulaması yaşam döngüsündeki her bir aşamayı ve katılımının kapsamını tanımlayan resmi prosedürler ve talimatlar olduğunda önemli ölçüde artar.

- Bilgi sistemleri denetçisi, iş uygulamasının geliştirme aşamaları boyunca uygulanan yöntem ve tekniklere ilişkin değerlendirme sağlayabilir.

- Bilgi sistemleri denetçisi, sistem geliştirme projesinin tüm ilgili alanlarını ve aşamalarını gözden geçirerek planlanan hedef ve organizasyonel prosedürlere bağlılık konusunda yönetime bağımsız olarak rapor verebilir.

- Bilgi sistemleri denetçisi, sistemin seçilmiş kısımlarını belirleyerek beceri ve yeteneklerine yönelik teknik yönlerini değerlendirebilir.

Bu yaklaşımın avantajları aşağıdaki gibidir:

- Kontrollü ve Yönetilebilir İlerleme: V Modeli, her aşamanın sonunda bir test aşamasını içerir ve her adım belirli bir sırayla gerçekleştirilir. Bu, projenin daha iyi kontrol edilmesini ve yönetilmesini sağlar.

- Daha İyi Kalite Güvencesi: Her aşama sonunda yapılan testler, hataların erken tespit edilmesine ve düzeltilmesine yardımcı olur, bu da genel ürün kalitesini artırır.

- Belirgin İş Akışı: V Modeli, projenin iş akışını net ve belirgin bir şekilde tanımlar, böylece herkesin ne yapması gerektiği açıktır.

- Dökümantasyonun Zenginliği: Her aşama, ayrıntılı dökümantasyon gerektirir, bu da projenin ilerlemesini ve gereksinimlerin izlenmesini kolaylaştırır.

- Risk Azaltma: Testler, projenin her aşamasında yapılacağı için, olası hatalar ve sorunlar daha erken tespit edilir ve bu da projenin sonraki aşamalarda daha az risk taşımasını sağlar.

Bu yaklaşımın dezavantajları aşağıdaki gibidir:

- Esneklik Eksikliği: V Modeli, değişikliklere karşı dirençlidir ve projenin ilerleyen aşamalarında değişiklikler maliyetli ve zaman alıcı olabilir.

- Müşteri Katılımı Sorunu: Müşteriler ürünü genellikle son test aşamasında görürler, bu nedenle müşteri geri bildirimini dikkate almak ve değişiklikleri yapmak zor olabilir.

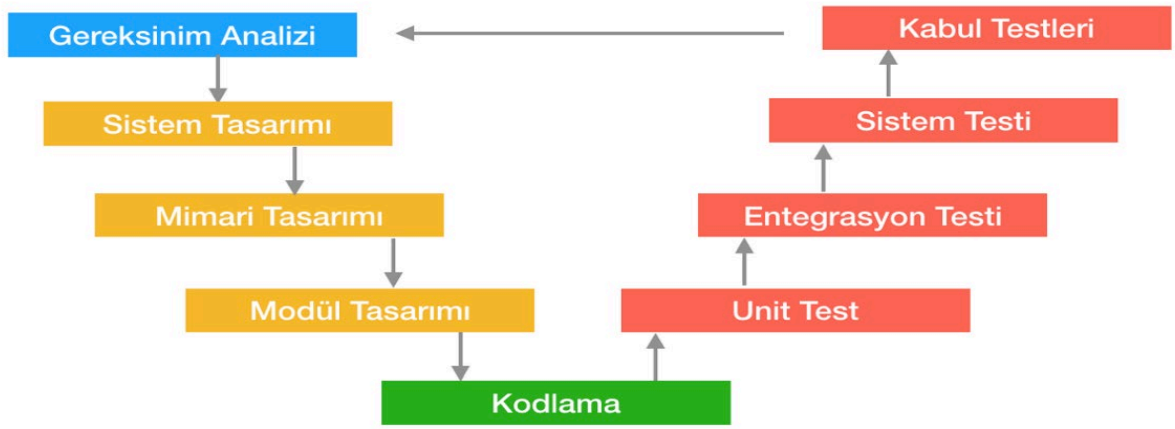
- Uzun Süreli Projeler İçin Uygun Değil: Büyük ve karmaşık projeler için uygun değildir, çünkü her aşama tamamlandıktan sonra sonuçlar görülemeyebilir ve değişiklik yapmak zor olabilir.

- Riskli Değişiklikler: Değişiklikler projenin ilerleyişini geriye döndürebilir ve maliyeti artırabilir. Bu nedenle, gereksinimlerin baştan net ve doğru bir şekilde belirlenmesi önemlidir.

- Yaratıcılık Kısıtlamaları: Model, yaratıcılık gerektiren projeler için uygun olmayabilir, çünkü çok katı bir yapıya sahiptir.

- Kullanım Alanı Sınırlıdır: V Modeli, özellikle büyük yazılım projeleri için uygun değildir ve bazen daha esnek metodolojilere ihtiyaç duyulabilir.

- Müşteriye Geç Teslimat Riski: Müşteriler ürünü projenin sonunda görürler, bu da müşteriye geç teslimat riskini artırabilir ve müşteri tatminini azaltabilir.



V Şekil Model Şeması

3) Yinelemeli (İteratif) Model

Proje yönetiminde, özellikle büyük projeler olmak üzere, projeleri safhalara bölmek ve sırasıyla bu safhaları gerçekleştirmek en doğal yöntem olarak görülür. Bu yöntemde, proje ürünü tüm safhalar tamamlandıktan sonra ortaya çıkar. Ancak bu yönteme alternatif bir yöntem olarak yinelemeli model geliştirilmiştir. Yinelemeli model, şelale modelinde görülen esnek olmayan proje disiplini yüzünden oluşan bir ihtiyaç sonucu ortaya çıkmış modeller arasındadır. Bu model, ilk denemede bir projeyi bitirip teslim etmek yani "*aceleyle bitirmek*" yerine, *"yavaş ve istikrarlı"* bir şekilde projeyi tamamlamayı teşvik eder. Görevlerin tekrar edilmesiyle proje kademeli olarak geliştirilir. Bu modelde, projenin müşteriye gösterilmesi ve dağıtımının gerçekleşmesi için projenin tamamlanması gerekmez. Bunun yerine proje küçük parçalara bölünür ve bu parçalar sonuca ulaştırılmak üzere geliştirilmeye başlanır. Projenin tanımı daha çok projenin geliştirilmesi aşamasında gerçekleşmektedir. Bu model bir nevi şelale ve çevik modellerinin bir karışımı olarak değerlendirilebilir.

Modelde bir döngü söz konusudur. Planlama aşaması ile başlar. Planlamalardan geçip bir döngüye girilir. İstek analizleri, analizler, tasarımlar yapılır ve uygulamaya geçilir. Uygulamadan sonra iki durum söz konusudur. Birincisi canlıya geçiş durumudur. Deployment edilir ve tabi problemler her aşamada olduğu gibi bu aşamada da yaşanır. Ama bu problemler testlerin birer parçası olarak görülebilir. Bir yandan da testler devam eder. Bu problemler sistemi besler. Testlerden ve canlıdan gelen bilgilere göre tekrar değerlendirilmeye tabii tutulur. Sonra tekrar istek analizler, canlıya bir bilgi verilmesi ve tekrar testler gibi suren bir döngü vardır. Bu sırada planlamada değişiklikler olabilir. Her yineleme sırasında, geliştirme süreci gereksinimlerden teste kadar her aşamadan geçer ve sonraki her döngü, süreci aşamalı olarak iyileştirir.

Bu model bütün bir projeyi geri dönülmez bir disiplin içerisinde geliştirmek yerine, parçalara ayırarak şelale modelini parçalarda uygulamayı tercih eder. Böylece şelale modelindeki öngörülemeyen hatalar riskini düşürmüş olur. Ancak bu yönü ile de esnekliğini kaybederek çevik modelden ayrılır. Talepleri toplanmış bir proje üç kısma bölünerek kurgulanır. Bu ayrımdan sonra ikinci aşama olan geliştirme aşamasına geçilir. Geliştirme aşamasında oluşum olarak ayrılan her kısım ayrı ayrı geliştirilerek testlere tabi tutulur. Testlerden sonra gerekli görülmesi halinde eksiklikler, uyarlama ve bakımlar ile sağlanır ve müşteriye teslim gerçekleşir. Yapı bakımından esneklik ve ihtiyaçlara sonradan cevap verebilme imkânına sahip olan bu proje modelinde yaşanan sorun, şelale modelinde olduğu gibi, ayrılmış kısımların tesliminde oluşacak problemlerde geri dönüş zorluğudur. Ancak bu zorluk bütün projenin yalnızca bir kısmına hitap ettiği için getireceği masraf klasik şelale modelinden çok daha az olacaktır. Bu özelliği ile riski ve maliyeti düşürebilse de, projeler için çevik modelde olduğu kadar geniş bir esneklik sağlamamaktadır. Sahip olduğu bu özelliği nedeni ile modelde eksik olan kısım daha sonra artırımlı model ile giderilmeye çalışılmıştır (Radack ve ark., 2009).

Bu yaklaşımın avantajları aşağıdaki gibidir:

- Esneklik: Yinelemeli model, projenin ilerleyen aşamalarında değişikliklere daha iyi uyum sağlar. İhtiyaçlar ve gereksinimler, her döngünün sonunda gözden geçirilebilir ve güncellenebilir.

- Erken Prototip Üretimi: İlk döngülerde çalışabilir prototipler üretilerek, müşterinin geri bildirimi daha erken aşamalarda alınabilir. Bu, müşteri memnuniyetini artırabilir.

- Risk Azaltma: Her döngü, daha küçük bir ölçekte bir projeyi temsil eder, bu da her aşamada olası hataların ve risklerin erken tespit edilmesine yardımcı olur.

- Sürekli İyileştirme: Her yineleme döngüsü, öncekinden öğrenilen bilgileri ve deneyimleri kullanarak sürecin iyileştirilmesine katkıda bulunur.

- Müşteri Katılımı: Müşteri, her döngünün sonunda sonuçları görür ve geri bildirim sağlar, bu da müşteri katılımını artırır.

Bu yaklaşımın dezavantajları aşağıdaki gibidir:

- Yüksek Yönetim İhtiyacı: Her döngü ayrı bir planlama, analiz ve tasarım gerektirir, bu da yönetim açısından daha fazla kaynak ve zaman gerektirir.

- Artan Karmaşıklık: Projenin ilerleyen döngülerinde, daha önceki döngülerin sonuçlarına uyum sağlama ve entegrasyon karmaşıklığı artabilir.

- Müşteri İlginin Azalması: Müşteri, her döngünün sonunda sonuçları gördüğü için proje tamamlandığını düşünebilir ve sonraki döngülere olan ilgisi azalabilir.

- Dökümantasyon Zorluğu: Her döngüde ayrı dökümantasyon gerektirir, bu da dokümantasyonun yönetimini karmaşıklaştırabilir.

- Kapsam Kontrolü Zorluğu: Her yineleme sonunda yeni özellikler eklenmesi veya gereksinimlerin değiştirilmesi riskini taşır, bu da kapsam kontrolünün zorlaşmasına neden olabilir.

- Uygun Olmayan Projeler İçin: Küçük ve orta ölçekli projeler için uygun olsa da, büyük ve karmaşık projeler için uygun olmayabilir. Bu tür projeler için daha fazla yönetim ve denetim gerekebilir.

2.1.1. Sistem Geliştirme Yaşam Döngüsü Yaklaşımı

SDLC yaklaşımı her biri farklı aktiviteler içeren ve birbirlerini izleyen aşamalardan oluşur. Her aşama için amaç ve hedefler, ulaşılmak istenen sonuçlar ve bu kapsamda belirlenmiş sorumluluklar vardır. SDLC yeni bir proje kapsamında işletme içi geliştirme yapmak için kullanılabildiği gibi üçüncü partiden sistem satın alındığı zaman da kullanılabilir. Bu durumda izlenecek fazlar da değişir.

SDLC'nin başarı faktörleri aşağıdakileri içerir:

- Müşteri Memnuniyeti: Projenin başarısı, müşterinin ihtiyaçlarını karşılayacak bir ürün veya hizmet sunulması ile ölçülür. Müşteriye sunulan çözümün beklentileri karşılaması ve onların memnuniyetini sağlaması büyük bir öneme sahiptir.

- Kalite: SDLC, ürün veya hizmetin kalitesini artırmak için farklı aşamalarda kalite kontrol ve güvence önlemlerini içerir. Kaliteli bir ürün veya hizmet, kullanıcılar arasında güvenilirliği artırır.

- İş Verimliliği: SDLC, iş süreçlerini optimize etmek ve verimliliği artırmak amacıyla tasarım ve geliştirme aşamalarında iyileştirmeler yapmayı içerir. Daha verimli iş süreçleri, maliyetleri düşürebilir ve zaman tasarrufu sağlayabilir.

- Ekonomik Değer Sağlama: Proje maliyetleri ile sağlanan değer arasındaki denge, SDLC'nin başarısını belirler. Projenin ekonomik olarak sürdürülebilir olması ve yatırım getirisinin beklentileri karşılaması önemlidir.

- Zamanında ve Bütçe İçinde Tamamlama: Proje, planlanan süre içinde ve tahmini bütçe çerçevesinde başarılı bir şekilde tamamlanmalıdır. Gecikmeler ve maliyet aşımları, projenin başarısızlıkla sonuçlanmasına yol açabilir.

- Uyumluluk ve Güvenlik: Ürün veya hizmetin, ilgili düzenlemelere ve güvenlik standartlarına uygun olması gereklidir. Veri güvenliği, gizlilik ve yasal uyumluluk, projenin başarısının kritik unsurlarıdır.

- İş Sürekliliği: Proje, iş sürekliliği planlarını ve risk yönetimini içermeli ve iş kesintilerini minimumda tutmalıdır. Özellikle kritik iş süreçlerini etkileyen projelerde bu faktör büyük öneme sahiptir. İş Sürekliliği planlamasında, DevOps (Geliştirme ve Operasyonlar Arası İşbirliği) ve Sürekli Entegrasyon (CI) yöntemlerinin entegrasyonu, projelerin iş sürekliliğini artırarak sistemin kesintiye uğramadan devam etmesini sağlar. Ayrıca, bulut tabanlı sistemler **ve** API (Uygulama Programlama Arayüzü) entegrasyonları ile sistemler daha esnek hale gelir, böylece iş süreçleri optimize edilir ve projelerin daha hızlı bir şekilde tamamlanması sağlanır.

Projelerde neden sistem geliştirme yaşam döngüsü (SDLC) kullanmalıyız?

- Yönetim ve Kontrol Sağlama: SDLC, projenin tüm aşamalarını planlama, izleme ve kontrol etme fırsatı sunar. Proje yöneticileri, her aşamada ilerlemeyi değerlendirebilir ve gerektiğinde düzeltici önlemler alabilir.

- Gereksinimleri Belirleme: Projenin başlangıcında gereksinimlerin ayrıntılı bir şekilde belirlenmesi, projenin amacının ve kapsamının net bir şekilde anlaşılmasını sağlar. Örnek: Bir müşteri yönetim sistemi geliştiriliyorsa, kullanıcıların hangi işlevleri bekledikleri SDLC aşamalarında belirlenir.

- Riskleri Azaltma: SDLC, projenin risklerini tanımlama ve yönetme fırsatı sunar. Örnek: İlerleyen aşamalarda oluşabilecek maliyet aşımları veya kaynak eksiklikleri önceden tanımlanabilir ve önlemler alınabilir.

- Kaliteyi ve Güvenilirliği Artırma: SDLC, kalite kontrol ve güvence süreçlerini içerir. Yazılımın ve sistemlerin daha güvenilir ve hatasız olmasını sağlar. Örnek: Bir finans uygulaması geliştirilirken, finansal işlemlerin doğru bir şekilde işlendiğini doğrulamak için testler yapılır.

- Dokümantasyon Oluşturma: Her aşamada oluşturulan dokümantasyon, projenin ilerleyişini ve gereksinimleri belgelemeye yardımcı olur. Bu, gelecekteki bakım ve güncelleme süreçlerini kolaylaştırır.

Projelerde sistem geliştirme yaşam döngüsü (SDLC) kullanmazsak ne olur?

- Kontrolsüz Projeler: SDLC kullanılmadığında, projeler kontrolsüz hale gelebilir. Bu, bütçe aşımları, süreç sorunları ve düşük kaliteli sonuçlarla sonuçlanabilir.

- Belirsizlik: Gereksinimlerin ve hedeflerin net olmaması, projenin ilerleyişini belirsizleştirir. Bu, projenin başarısızlığına yol açabilir.

- Risklerin Yönetilememesi: Projedeki risklerin önceden tanımlanmaması ve yönetilmemesi, beklenmeyen sorunlara yol açabilir ve projeyi tehlikeye atabilir.

- Kalite Sorunları: Kalite kontrol ve güvence eksikliği, yazılım veya sistemlerin hatalarla dolu olmasına ve kullanıcı memnuniyetsizliğine yol açabilir.

- Müşteri Memnuniyetsizliği: Gereksinimlerin ve beklentilerin belirsiz olduğu projeler, müşteri memnuniyetsizliğine yol açabilir ve müşterinin projeyi kabul etmemesi riskini artırabilir.

- Kötü İş Süreçleri: SDLC olmadan, iş süreçleri belirsiz ve düzensiz olabilir. Bu, projenin verimliliğini düşürebilir ve proje hedeflerine ulaşmayı zorlaştırabilir.

- Karmaşık Sorunların Artması: Gereksinimlerin ve tasarımın net olmaması, projenin ilerleyen aşamalarında karmaşık sorunların ortaya çıkmasına neden olabilir. Bu da projenin gecikmesine yol açabilir.

- Kaynak İsrafı: Proje kaynakları, gereksinimlerin belirsizliği nedeniyle etkin bir şekilde kullanılamaz. Bu, bütçe aşımlarına ve kaynak israfına yol açabilir.

- Proje Başarısızlığı: Kontrolsüz projelerin, hedeflere ulaşma olasılığı daha düşüktür. Proje başarısızlığı riski artar ve proje sonuçlanmamış veya terk edilmiş olabilir.

- Müşteri Kaybı: Müşteriler, belirsizlikler nedeniyle projenin tamamlanmaması veya düşük kaliteli sonuçlar nedeniyle kaybedilebilir.

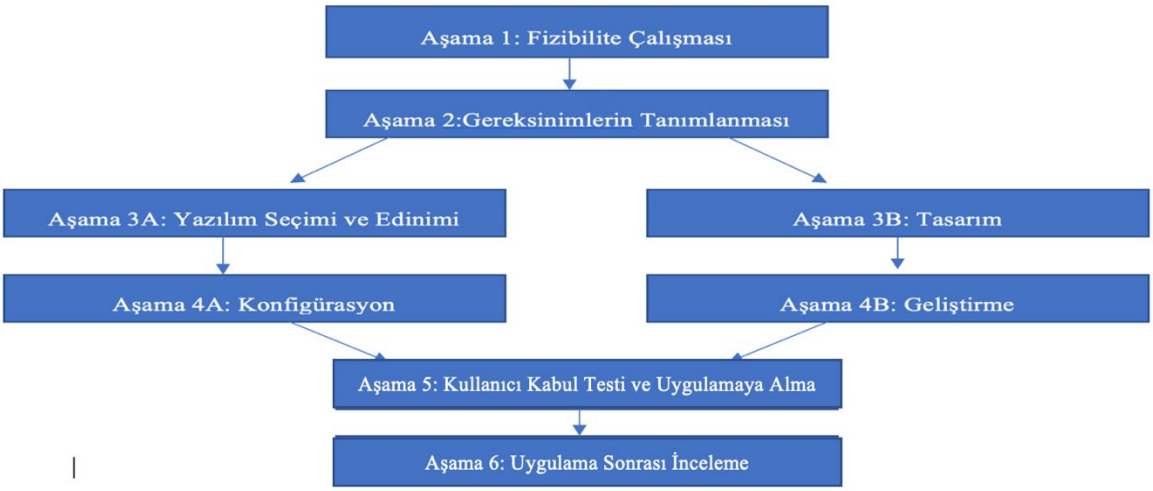
2.1.2. Sistem Geliştirme Yaşam Döngüsü Yaklaşımının Aşamaları

SDLC kapsamında her bir aşama, bu aşamaların amacı, süreçteki diğer aşamalar ile ilişkisi, aşama kapsamında gerçekleştirilecek faaliyetler tamamlandığında beklenen sonuçlar açısından detaylandırılmıştır.

SDLC; bilgisayar endüstrisinde, yeni yazılım ve donanımların oluşturulmasını, oluşturulan yazılım ve donanımların kullanılmasını, tamamlanan ürünlerin elden çıkarılması sürecini tanımlar, düzenler ve yönetilmesini sağlar.

SDLC'nin bir fizibilite çalışmasıyla başlayan ve gereksinimlerin tanımlanması, tasarımı, geliştirilmesi, uygulanması ve uygulama sonrası boyunca ilerleyen yazılım geliştirmeye yönelik sistematik, sıralı bir yaklaşım ile devam eder. Ayrıca, bilgi sistemleri denetçisi, SDLC aşamalarını iyi bir şekilde bilmeli ve fizibilite çalışmasını detaylı olarak değerlendirebilmelidir.

SDLC'nin her aşamasında, yapay zeka (AI) ve makine öğrenmesi (ML) yöntemlerinin entegrasyonu, yazılım ve sistem geliştirme süreçlerinde daha hızlı veri analizi ve hata tespiti yapılmasına olanak tanır. Bu, her aşamanın çıktılarının daha verimli ve doğru bir şekilde değerlendirilmesini sağlar.



SDCL'nin Aşamaları Şeması

Aşama 1-Fizibilite Çalışması:

Fizibilite çalışması, bir proje yapmaya karar verme süreci boyunca hem teknik hem finansal açılardan gerekli araştırmalar ile değerlendirme yapılmasıdır. Olası problemler ve tehditler araştırılarak yatırımın istenilen hedefe ulaşıp ulaşamayacağı, kar getirip getiremeyeceği öngörülmeye çalışılır. Fizibilite çalışması ile sistemin üretkenlik kazanımlarında veya gelecekteki maliyetlerden kaçınmada uygulanmasının stratejik faydalarını belirlemek, yeni bir sistem maliyet tasarruflarını ve miktarını belirlemek ve sistem uygulanmasında ortaya çıkan maliyetler için bir geri ödeme planı tahmin etmek. Ayrıca iş kullanıcılarının hazır bulunmaları ve iş süreçlerinin olgunluğu gibi somut olmayan faktörleri de göz önüne alacak ve değerlendirilecektir. Bu iş olurluk incelemesi bir sonraki aşamaya geçmenin gerekçesini sağlar.

Fizibilite çalışmasında, bulut bilişim (cloud computing) çözümlerinin, sistem esnekliğini artırması ve düşük maliyetle yüksek ölçeklenebilirlik sağlaması nedeniyle önemli bir rolü vardır. Ayrıca, projelerde veri analitiği kullanılarak, iş süreçlerinin performansı ve verimliliği hakkında daha doğru tahminler yapılabilir.

Fizibilite aşamasında, blockchain (blok zinciri) gibi yeni nesil teknolojiler de değerlendirilmelidir. Bu teknolojiler, veri güvenliğini artırır ve şeffaflık sağlar. Ayrıca, siber güvenlik

unsurları ve olası tehditler üzerinde de durulmalı, projeye dair güvenlik önlemleri bu aşamada belirlenmelidir.

Fizibilite çalışmasında aşağıdaki aşamaların gerçekleştirilmesi önemlidir:

- İşletmenin hedefleri ve eğer sorun ile ilgili ise sorunun incelenmesi ve kapsamının belirlenmesi,

- Proje kapsamının tanımlama,

- Mevcut durum analizi yapma,

- Paydaş ihtiyaçlarına dayalı gereksinimleri belirleme,

- Önerilen bir yaklaşım sağlama,

- Yaklaşımın maliyet etkinliğini değerlendirme,

- Paydaşlarla resmi bir gözden geçirme yapma.

Fizibilite çalışması sırasında yapılacak hususlar:

- Çözüme ulaşmak için gereken zaman çerçevesi belirlenir.

- İş gereksinimlerinin karşılanabilmesi için en iyi risk tabanlı çözüm belirlenir. Sonuçta süreç kolayca SDLC ya da hızlı uygulama geliştirme/(RAD) ile eşleştirilebilir.

- Mevcut sistemde küçük bir değişiklik ya da geçici çözüm ile çözüm bulunup bulunamayacağı belirlenir.

- Üçüncü partiden alınacak bir ürünün çözüm bulup bulmayacağı belirlenir.

- Çözüm için gerekli maliyet belirlenir.

- Çözümün iş stratejisine uyumu değerlendirilir.

Fizibilite çalışması sırasında, bilgi sistemleri denetçisi aşağıdakileri yapmalıdır:

- İhtiyacın Kritikliğini Belirleme: Bilgi sistemleri denetçisi, projenin veya çözümün işletme için ne kadar kritik olduğunu değerlendirmelidir. Hangi ihtiyaçların acil ve önemli olduğunu belirlemek, projenin önceliklerini saptamak için önemlidir.

- Çözüm Elde Edilebilirliği Değerlendirme: Mevcut sistemlerin veya kaynakların kullanılmasıyla sorunun çözülebileceği bir senaryo varsa, bu alternatif yaklaşımın uygunluğunu değerlendirmelidir. Eğer mevcut kaynaklarla çözüm elde edilemiyorsa, alternatif çözümlerin maliyet ve fayda analizini yapmalıdır.

- Çıktıların Gözden Geçirilmesi: Fizibilite çalışması sırasında üretilen tüm belgeler, raporlar ve analizler bilgi sistemleri denetçisi tarafından gözden geçirilmelidir. Bu belgelerin doğruluğu, eksiksizliği ve uygunluğu değerlendirilmelidir.

- Maliyet ve Fayda Analizi: Projeye ilişkin maliyetlerin ve beklenen faydaların gerçekçi ve doğru bir şekilde değerlendirilmesi gereklidir. Bilgi sistemleri denetçisi, projenin bütçesinin ve maliyetlerin kontrol edilmesine katkıda bulunmalıdır. Aynı zamanda beklenen faydaların, projenin maliyetlerini karşılayıp karşılayamayacağını analiz etmelidir.

- Çözümün Uygunluğu: Önerilen çözümün işletme ihtiyaçlarına ve stratejilerine uygunluğunu değerlendirmelidir. Çözümün iş süreçleri ile nasıl entegre olacağını ve işletmeye nasıl katkı sağlayacağını anlamak önemlidir.

- Risk Değerlendirmesi: Bilgi sistemleri denetçisi, projenin teknik ve finansal risklerini belirlemeli ve bunların olası etkilerini değerlendirmelidir. Bu risklerin erken tespiti ve etkili bir şekilde yönetilmesi, projenin başarısını büyük ölçüde etkileyebilir.

- Teknolojik Değerlendirme: Önerilen çözümün teknik olarak uygulanabilirliğini incelemeli ve hangi teknolojilerin kullanılacağını, donanım gereksinimlerini ve yazılım altyapısını analiz etmelidir.

- Müşteri İletişimi: Bilgi sistemleri denetçisi, müşteri veya işletme temsilcileri ile düzenli iletişimde bulunmalı ve onların projenin her aşamasında geri bildirimlerini almalıdır. Müşteri beklentilerini anlamak ve projenin bu beklentilere uygun olduğundan emin olmak önemlidir.

- Hukuki ve Yönetmelik Değerlendirmesi: Bilgi sistemleri denetçisi, projeyi etkileyebilecek hukuki veya düzenleyici gereklilikleri değerlendirmelidir. Bu, veri gizliliği, güvenlik, fikri mülkiyet hakları ve diğer yasal konuları içerebilir.

- İş Sürekliliği Planlaması: Proje sırasında veya sonrasında oluşabilecek kesintilere karşı iş sürekliliği planlarını gözden geçirmeli ve gerektiğinde güncellemelidir. İş sürekliliği, projenin başarıyla tamamlanmasını sağlamak için önemlidir.

- Çevresel ve Sosyal Etkilerin Değerlendirmesi: Bilgi sistemleri denetçisi, projenin çevresel ve sosyal etkilerini değerlendirmelidir. Bu, projenin çevresel sürdürülebilirlik ve toplumsal sorumluluk açısından uygunluğunun değerlendirilmesini içerebilir.

İş Olurluğu (Business Case)

Bir iş olurluğu sonucunda, bir işletmenin bir projenin devam edip etmemesi gerektiğine karar vermesi için gereken bilgiler sağlanır. Bir projedeki tamam ya da devam kararının alındığı adımdır. Maliyetlerin ve iş faydalarının karşılaştırılması sonucunda bir projenin kurulması veya sürdürülmesi için gerekçe sağlar. Uygun olmayan projeler bu aşamada sonlandırılabilir.

İş olurluğunun sağladığı bilgiler, işletme yöneticilerine ve paydaşlara projenin değerini ve yatırımın geri dönüşünü anlama ve karar verme konusunda rehberlik eder. İş olurluğu, aşağıdaki nedenlerle kullanılmalıdır:

- Karar Alma Sürecini Destekler: İş olurluğu, bir projenin başlaması, sürdürülmesi veya sonlandırılması kararını etkileyecek temel bilgileri sağlar. İşletme yöneticileri bu bilgilere dayanarak projenin stratejik hedeflere katkı sağlayıp sağlamayacağına karar verirler.

- Kaynak Yönetimini İyileştirir: İş olurluğu, bir projenin maliyetlerini ve faydalarını açıkça ortaya koyar. Bu sayede işletme, kaynakları en iyi şekilde tahsis edebilir ve bütçe kontrolünü sağlayabilir. Yeni nesil teknolojiler ve veri analitiği, iş olurluğu analizlerinde önemli bir yer tutar. Özellikle yapay zeka (AI) ve makine öğrenmesi (ML), projelerin risklerini daha doğru tahmin edebilir ve maliyet-fayda analizlerini optimize edebilir. Ayrıca, bulut bilişim (cloud computing) altyapıları ile veri depolama ve işlem kapasitesi artırılarak, projelerin daha esnek ve sürdürülebilir hale gelmesi sağlanabilir. Proje olurluğu aşamasında blok zinciri (blockchain) teknolojisi, veri güvenliği ve şeffaflık sağlar, böylece projenin yasal gereksinimlere uygunluğu da artırılabilir. Bu tür teknolojiler sayesinde proje paydaşlarının güvenliği ve veri gizliliği de sağlanmış olur.

- Riskleri Değerlendirir: İş olurluğu, projenin getirebileceği riskleri tanımlar ve değerlendirir. Bu, işletmenin riskleri minimize etmek veya yönetmek için stratejiler geliştirmesine olanak tanır.

- Yatırım Geri Dönüşünü Değerlendirir: İş olurluğu, projenin maliyetleri ve beklenen faydaları karşılaştırarak yatırımın geri dönüş süresini ve getirisini hesaplar. Bu, işletmenin uzun vadeli stratejik planlarını destekler.

- Proje Başarı Kriterlerini Belirler: İş olurluğu, projenin başarı kriterlerini belirlemeye yardımcı olur. Bu kriterler, projenin hedeflerine ne kadar yaklaştığını ve başarılı olup olmadığını değerlendirmek için kullanılır.

- Paydaşları Bilgilendirir: İş olurluğu, proje paydaşlarına projenin neden gerekli olduğunu, ne beklenildiğini ve nasıl değerlendirileceğini açıklar. Bu, proje ekibi, finans departmanı, yöneticiler ve diğer ilgili taraflar arasında iletişimi kolaylaştırır.

- Alternatif Çözümleri İnceleme Fırsatı Sunar: İş olurluğu aynı zamanda alternatif çözümleri değerlendirmek için bir fırsat sunar. Birden fazla proje veya yaklaşım arasında karşılaştırma yapılabilir ve en uygun olanı seçmek için verilere dayalı bir temel oluşturulabilir.

Aşama 2- Gereksinimlerin Tanımlanması:

Gereksinim analizi, projede hayati bir rol oynar. Çünkü projenin başarısı ve başarısızlığı gereksinim analizine bağlıdır. Projeye özel ihtiyacın ve çözüm gerektiren sorunun gereksinimler ile tanımlanır. Bu tanımlama sırasında kullanıcının katılımı çok önemlidir. Gerek özelleşmiş bir yaklaşım gerekse tanımlanmış ve belgelenmiş bir tedarik sürecini takiben satıcı tarafından sağlanan yazılım paketinde olsun kullanıcıların aktif olarak dahil olması beklenir.

Gereksinim tanımları, bir sistemin ne yapması gerektiği, kullanıcıların bir sistemle nasıl etkileşime gireceği, sistemin hangi koşullarda işleyeceği ve sistemin yerine getirmesi gereken bilgi kriterleri hakkında açıklamalar içermelidir.

Gereksinim tanımlarını başarıyla tamamlamak için proje ekibi aşağıdaki gibi görevleri yerine getirecektir:

- Paydaşları belirlemek,

- Gereksinimleri yapısal bir formatta kaydetmek ve paydaşlara danışmak,

- Gereksinimlerin eksiksiz, tutarlı, açık, doğrulanabilir, değiştirilebilir, test edilebilir ve izlenebilir olduğunu doğrulamak,

- Gereksinimleri ile beklentiler arasındaki farkları tespit etmek ve düzeltmek,

- Sistemsel sınırlamaları belirlemek,

- İlgili güvenlik gereksinimlerini belirlemek,

- Kullanıcı gereksinimlerini sistemsel gereksinimlere dönüştürmek (Bu aşamada prototip bir kullanıcı arayüzü bu aşamada hazırlanabilir.),

- Gereksinimlerin tarihsel bir şekilde saklanabilmesini sağlamak (Bu amaçla hazırlanmış birçok paket program temin edilebilir.),

- Gereksinimler kapsamında herhangi bir kısıtlamayı belirlemek,

- Gereksinimlerin eksiksiz, tutarlı açık doğrulanabilir, değiştirilebilir ve izlenebilir olduğunu doğrulamak (Bu şekilde etkin bir iş olurluğu takibi yapılabilir.),

- Proje paydaşları arasındaki uyuşmazlıkları (çatışmaları) çözmek,

- Gereksinimler ve kaynaklar arasındaki uyumsuzlukları çözmek.

Gereksinimlerin doğru bir şekilde tanımlanması için özellikle kullanıcı deneyimi (UX) tasarımının (User Experience Design) ve *siber güvenlik* (Cybersecurity) gereksinimlerinin ele alınması büyük önem taşır. Gereksinim tanımları sırasında, özellikle bulut tabanlı sistemlerin kullanımıyla birlikte veri güvenliği ve erişilebilirliği sağlanmalı, ayrıca projeye dair kritik iş süreçlerinin güvenliği ön planda tutulmalıdır. Ayrıca, projede kullanılacak yazılım altyapısının ve donanım gereksinimlerinin belirlenmesi süreci, gereksinimlerin doğruluğunu sağlamak adına önemli bir adımdır. Bu tür önlemlerle projenin esnekliği ve güvenliği artırılabilir.

Gereksinimlerin tanımlanması aşamasında birçok hata ve sorunla karşılaşılabilir. Bu hatalar, projenin ilerlemesini olumsuz etkileyebilir ve projenin başarısızlığına yol açabilir. Gereksinimlerin tanımlanması aşamasında sıkça yapılan hatalar ve neden oldukları olumsuz durumlar aşağıda maddeler halinde açıklanmıştır:

- Eksik veya Belirsiz Gereksinimler: Gereksinimlerin eksik veya belirsiz bir şekilde tanımlanması, projenin ilerleyen aşamalarında karmaşıklığa ve hatalara yol açabilir. Eksik veya belirsiz gereksinimler, proje ekibinin ne yapması gerektiğini tam olarak anlamasını zorlaştırır.

- Aşırı Detaylı Gereksinimler: Gereksinimlerin aşırı detaylı olması, projeyi karmaşık hale getirebilir ve gereksiz maliyetlere yol açabilir. Aşırı detaylı gereksinimler, projenin esnekliğini azaltabilir ve ilerleme sürecini yavaşlatabilir.

- Gereksinimlerin Değişkenliği: Proje sürecinde gereksinimlerin sık sık değişmesi veya güncellenmesi, proje ekibini zorlayabilir ve maliyetleri artırabilir. Gereksinimlerin sabitlenmemesi, projenin sürekli olarak yeniden değerlendirilmesine neden olabilir.

- Paydaşların Yetersiz Katılımı: Projenin paydaşları, gereksinimlerin tanımlanması sürecine yeterince dahil edilmezse, projenin sonuçları kullanıcı ihtiyaçlarına uygun olmayabilir. Kullanıcıların beklentilerini anlamak ve gereksinimleri doğru bir şekilde tanımlamak için paydaşların aktif katılımı gereklidir.

- Gereksinimlerin İzlenmemesi: Gereksinimlerin zaman içinde nasıl değiştiğini izlememek, projenin gereksinimlere uygunluğunu tehdit edebilir. Gereksinimlerin izlenmemesi, projenin hedeflerine ulaşmasını zorlaştırabilir.

- Gereksinimlerin Çatışması: Farklı paydaşlar arasında gereksinimlerin çatışması durumunda, projenin ilerlemesi engellenebilir. Bu çatışmaların çözülmesi ve uygun bir denge bulunması önemlidir.

- Gereksinimlerin Değerlendirilmemesi: Gereksinimlerin maliyet ve faydalarının dikkate alınmaması, projenin bütçesini aşmasına veya beklenen sonuçları sağlayamamasına neden olabilir.

- Kullanıcı Deneyiminin İhmal Edilmesi: Kullanıcıların projenin sonuçlarıyla nasıl etkileşimde bulunacakları ve kullanacakları göz ardı edilirse, projenin kullanıcı kabulünü ve başarısını olumsuz etkileyebilir.

- İletişim Eksikliği: Proje ekibi üyeleri ve paydaşlar arasında yetersiz iletişim, gereksinimlerin yanlış anlaşılmasına veya eksik bilgiye yol açabilir. İyi bir iletişim stratejisi gereklidir.

Bu hataların önlenmesi veya düzeltilmesi için gereksinim tanımlama sürecine özen göstermek, proje yönetiminde başarıyı artırabilir ve olumsuz sonuçların önüne geçebilir.

Proje ekibi, gereksinimlerin doğru bir şekilde tanımlanmasının yanı sıra, gereksinimlerin teknik boyutunun da doğru bir şekilde ele alındığı teknik analiz ve sistem güvenliği incelemelerini yapmalıdır. Özellikle, kullanılan yazılımlar ve sistemlerin güvenliğini sağlayacak yöntemler belirlenmelidir. Ayrıca, gereksinimlerin teknoloji ve süreç değişikliklerine uyum sağlaması için çevik yöntemler (Agile methodologies) ve sürekli entegrasyon (CI) gibi modern yazılım geliştirme yaklaşımlarının uygulanması gerekebilir. Bu süreçte doğru bir yazılım geliştirme metodolojisi seçmek, gereksinimlerin sistematik bir şekilde ele alınmasını sağlayarak projenin başarısını artırabilir.

Bu adımda, iş analizi, teknik analiz ve güvenlik analizi yapılır. Bu analizlerin kapsamı kısaca aşağıda açıklanmaktadır.

İş Analizi

İş analizi, işlerin doğru, etkin ve sağlıklı biçimde değerlendirilmesi amacıyla işletmede yer alan her işin ayrı ayrı niteliğini, niceliğini, gereklerini, sorumluluklarını ve çalışma koşullarını bilimsel yöntemlerle inceleyen ve bilgi toplayan bir tekniktir. İşlerin nasıl yapılacağı değil, nasıl yapıldığını, ayrıntılarını ve çevresel durumunu ortaya koyar. İş gereksinimleri analizi olarak da adlandırılır. İş ve BT arasında bir köprü olacak şekilde konumlandırılır. İşin sınırlarının kesin olarak belirlendiği süreçtir. İş gereksinimleri analizi şunları içerir:

- İş Süreçlerinin Anlaşılması: İş Analizi, işletmenin mevcut iş süreçlerini ve operasyonlarını anlamayı içerir. Hangi görevlerin nasıl gerçekleştirildiği, iş süreçlerinin akışı ve işletmenin genel işleyişi bu aşamada incelenir.

- Kullanıcıların İhtiyaçlarının Tanımlanması: İş Analizi, işletmedeki kullanıcıların (örneğin, çalışanlar, müşteriler, yöneticiler) ihtiyaçlarını anlamayı hedefler. Bu ihtiyaçlar, kullanıcıların işlerini daha etkili bir şekilde yapabilmeleri için ne tür bir sisteme ihtiyaç duyduklarını belirlemek amacıyla incelenir.

- Sorunların ve Zorlukların Belirlenmesi: İş Analizi, mevcut iş süreçlerindeki sorunları ve zorlukları tespit etmeyi amaçlar. Bu sorunlar, projenin neden gereksinim duyulduğunu ve nasıl bir çözüm sunması gerektiğini anlamamıza yardımcı olur.

- İş ve BT Arasındaki Köprü: İş Analizi, işletmenin iş ihtiyaçlarını ve BT'nin teknik yeteneklerini birleştirmeyi amaçlar. İş ve BT arasındaki bu köprü, gereksinimlerin iş süreçlerine nasıl entegre edileceğini ve işletmenin hedeflerine nasıl katkı sağlayacağını belirler.

- Kapsamlı Bir Gereksinim Tanımlama: İş Analizi sonucunda, işletmenin iş süreçlerine ve kullanıcı ihtiyaçlarına uygun kapsamlı bir gereksinim listesi oluşturulur. Bu gereksinimler, proje tasarımı ve geliştirmesi için temel bir çerçeve sağlar.

İş Analizi, projelerde kullanılan yazılım geliştirme yöntemlerinin doğru bir şekilde seçilmesi için kritik bir adımdır. Bu bağlamda, iş süreçlerinin analiz edilmesi sadece mevcut durumu iyileştirmekle kalmaz, aynı zamanda süreçlerin çevik metodolojiler (Agile methodologies) ile nasıl daha verimli hale getirilebileceğini de araştırır. Çevik yazılım geliştirme (Agile software development) süreçlerinin, hızlı geri bildirim almayı, değişikliklere hızlı uyum sağlamayı ve kullanıcı ihtiyaçlarına göre esneklik sunmayı sağladığı bilinir. Ayrıca, iş analizi sırasında sürekli entegrasyon (CI) ve devops (yazılım geliştirme ve operasyonlarının entegrasyonu) gibi modern yazılım geliştirme yaklaşımlarının da göz önünde bulundurulması, projelerin daha hızlı ve verimli bir şekilde hayata geçmesine olanak sağlar. İş süreçleri ile teknoloji arasındaki köprünün bu şekilde oluşturulması, işletme hedeflerinin daha hızlı ve verimli bir şekilde ulaşılmasını destekler.

Teknik Analiz

Teknik gereksinimler analizi olarak da değerlendirilir. Teknik dizayn gereksinimleri bu aşamada ele alınır. İş gereksinimleri analizine göre teknik ihtiyaçların belirlendiği süreçtir. Teknik gereksinimler analizi şunları içerir:

- Teknik Altyapının İncelenmesi: Bu aşamada mevcut teknik altyapı ve sistemler detaylı bir şekilde incelenir. Var olan teknik bileşenler, donanım, yazılım ve veri tabanları gözden geçirilir.

- Teknolojik Gereksinimlerin Belirlenmesi: Hangi teknolojilerin kullanılacağı, hangi programlama dillerinin ve platformların tercih edileceği gibi teknik konular bu aşamada ele alınır.

- Veri Yönetimi ve Veri Akışının Tanımlanması: Projede kullanılacak verilerin nasıl saklanacağı, yönetileceği ve işleneceği konuları bu aşamada değerlendirilir. Veri akış diyagramları oluşturulabilir.

- Güvenlik Gereksinimlerinin Belirlenmesi: Projede kullanılacak sistemlerin güvenliği büyük bir önem taşır. Bu aşamada güvenlik gereksinimleri ve güvenlik önlemleri planlanır.

- Performans Gereksinimlerinin Tanımlanması: Sistem performansıyla ilgili gereksinimler bu aşamada belirlenir. Örneğin, sistem yanıt süreleri, kullanıcı sayısı ve eş zamanlı işlem kapasitesi gibi faktörler ele alınır.

- Entegrasyon Gereksinimlerinin Belirlenmesi: Projede kullanılacak farklı sistemlerin birbirleriyle nasıl entegre edileceği, veri paylaşımı ve iletişimi gibi konular bu aşamada incelenir.

- Teknik Dokümantasyonun Hazırlanması: Teknik gereksinimler ve tasarım belgeleri bu aşamada hazırlanır. Bu dokümantasyon, proje ekibi ve geliştiriciler için rehber niteliğindedir.

Teknik analiz aşamasında kullanılan yeni teknolojiler, yazılım projelerinin hızla gelişen gereksinimlerine uyum sağlamaya yardımcı olur. Özellikle, DevOps (Development Operations) yaklaşımı, yazılım geliştirme ve operasyon süreçlerini birleştirerek sürekli entegrasyonu (CI) ve sürekli teslimatı (CD) teşvik eder. Bu sayede yazılımın daha hızlı ve güvenilir bir şekilde geliştirilmesi sağlanır. Ayrıca, Yapay Zeka (AI) ve Makine Öğrenmesi (ML) destekli araçlar, teknik analiz sürecinde veri analizi ve karar verme süreçlerini iyileştirir. Bu teknolojiler, proje ekibine daha doğru ve verimli kararlar almayı sağlar ve projenin genel başarısını artırır.

Teknik Analiz, projenin teknik yönünü ve gereksinimlerini netleştirir ve bu sayede projenin başarıyla uygulanması için temel bir çerçeve sağlar. Bu aşama, projenin teknik detaylarının ve gereksinimlerinin anlaşılmasını sağlar ve geliştirme sürecinin planlanmasına yardımcı olur.

Güvenlik Analizi

Güvenlik gereksinimleri analizi olarak da ifade edilmektedir. Proje boyunca güvenlik kapsamında güvence verebilmek için gereklilikler bu kapsamda değerlendirilir. Tipik olarak, bilgi sistemleri denetçisi, aşağıdaki hususların gereksinim analizi kapsamında yeterliliğini gözden geçirmelidir:

- Çözümün düzenleyici, yasalara ve yasal gerekliliklere uygun olduğunu,

- Denetim izlerinin yeterli açıklıkta sistemin bir parçası olarak yer alıp almadığını (sorunların bu denetim izleri ile takip edilmesine imkân verilip vermeyeceği),

- Proje dahilinde veri oluşturma işleme, depolama, iletim ve imha süreçlerine kadar verilerin geçerliliği gizliliği bütünlüğü ve erişilebilirliği ile ilgili kontrollerin mevcut olduğunu,

- İşletmenin sistem güvenliği ekibinin, iş uygulaması içindeki veri yaşam döngüsü boyunca güvenlik kontrollerinin geliştirilmesinde ne derece yer aldığını.

Bu adımların yanı sıra güvenlik analizi sırasında, projenin risk değerlendirmesi yapılmalıdır. Potansiyel güvenlik tehditleri ve riskler tanımlanmalı, bu tehditlere karşı alınacak güvenlik önlemleri belirlenmelidir. Ayrıca güvenlik analizi, projenin yaşam döngüsü boyunca sürekli olarak güncel tutulmalı ve değişen tehditlere ve gereksinimlere uyum sağlamalıdır.

Bu aşamada, genel olarak aşağıdaki adımlar gerçekleştirilir:

- Sistemin fonksiyonel gereksinimleri, müşteri gereksinimleri, geliştirilecek sistem ile ilgili standartlar, yönetmelikler ile oluşan gereksinimler dökümante edilir.

- Müşteri gereksinimleri ile sistem arasındaki izlenebilirlik oluşturulur.

- Gereksinimlerin doğrulanması ve öncelik nitelikleri belirlenir.

- Kalite gereksinimleri belirlenir.

- Gereksinimler gözden geçirme sürecine uygun olarak gözden geçirilir.

Güvenlik analizi, proje yaşam döngüsünün her aşamasında aktif olarak güncellenmeli ve değişen tehditlere karşı sürekli uyum sağlamalıdır. Ayrıca, günümüzde Sıfır Güven (Zero Trust) modelinin uygulanması, güvenlik önlemleri açısından önemli bir gelişme olarak karşımıza çıkmaktadır. Bu modelde, hiçbir kullanıcıya ve cihaza güvensiz olarak erişim izni verilmez, her erişim için sürekli doğrulama yapılır. Bununla birlikte, şifreleme teknolojileri veri güvenliğinin sağlanmasında kritik rol oynar. Şifreleme, verilerin izinsiz erişimlere karşı korunmasını sağlar ve güvenlik açığı oluşturmaz. Ayrıca, gelişmiş tehdit izleme ve analiz araçları (Advanced Threat Detection and Analytics Tools), sistemlerdeki anormallikleri tespit etmek ve tehditlere anında müdahale etmek için önemlidir. Bu teknolojiler, güvenlik açıklarının erken tespit edilmesini sağlar ve potansiyel saldırılara karşı hızlı tepki verilmesini mümkün kılar.

Aşama 3A- Yazılım Seçimi ve Edinimi (Satın Alınan Sistemler):

Yazılım edinimi aslında SDLC'de bir aşama olarak değerlendirilmez. Yazılım geliştirme yerine edinme kararına varılırsa, gereksinimlerin belirlenmesinden sonra bu süreç izlenir. Bu süreçte bir satıcıya bağlı olmak yerine yeni sistem üzerindeki tam sahiplik ve kontrol sahibi olmanın faydaları da düşünülerek yazılımın temin edilmesi ön planda tutulur. Yazılım ediniminin avantajları ve dezavantajları vardır. Bunlara aşağıda yer verilmektedir.

Avantajlar:

- Hızlı Uygulama: Hazır yazılım, hızlı bir şekilde kurulabilir ve kullanılabilir, özel yazılım geliştirmeye göre daha kısa bir uygulama süresi sunar.

- Maliyet Etkinliği: Hazır yazılımların genellikle daha düşük başlangıç maliyetleri vardır ve özel yazılım geliştirmenin getirdiği yüksek maliyetlerden kaçınılabilir.

- Daha Az Kaynak Gereksinimi: Hazır yazılımın bakımı ve güncellenmesi genellikle sağlayıcı tarafından yapılır, bu da organizasyonun daha az IT kaynağına ihtiyaç duymasını sağlar.

- Öğrenme Eğrisi: Hazır yazılımlar genellikle kullanıcı dostu arayüzlere sahiptir, bu da personelin daha hızlı bir şekilde kullanmaya başlamasını sağlar.

- Yeniliklerden Yararlanma: Yazılım sağlayıcıları, yeni teknolojileri ve güncellemeleri hızlı bir şekilde uygular, bu da organizasyonun teknolojik olarak güncel kalmasına yardımcı olabilir.

- Topluluk ve Destek: Popüler yazılımlar genellikle büyük bir kullanıcı topluluğuna sahiptir, bu da kullanıcıların birbirlerine yardımcı olabileceği ve sorunları birlikte çözebileceği bir kaynak sunar.

Dezavantajlar:

- Özelleştirme Zorluğu: Hazır yazılımların bazı bölümleri özelleştirilebilir olsa da, özel gereksinimleri tam olarak karşılamak için sınırlıdır.

- Lisans ve Abonelik Maliyetleri: Kaliteli yazılımlar için yüksek lisans veya abonelik ücretleri ödenmesi gerekebilir, bu da uzun vadede maliyetli olabilir.

- Veri Aktarımı ve Entegrasyon Zorlukları: Mevcut sistemlerle yazılımın entegrasyonu ve veri aktarımı karmaşık olabilir.

- Güncelleme Kontrolü: Yazılım sağlayıcılarının güncelleme zamanlamaları organizasyonun iş akışını etkileyebilir, beklenmedik güncellemeler sorunlara yol açabilir.

- Veri Güvenliği Endişeleri: Üçüncü taraf yazılım sağlayıcılarına hassas verilere erişme izni vermek, veri güvenliği endişelerine yol açabilir.

Gereksinimlerin tamamlanması sonrasında bu gereksinimleri karşılamak için, tedarikçilerden veya çözüm sağlayıcılardan sağlanacak sistemler için, potansiyel tedarikçilerden teklif almak amacı ile işletme gereksinimlerini özetleyen bir teklif talebi oluşturulması gerekir.

Bu aşamada en önemli unsur, bir teklif talebi (RFP) hazırlamak ile başlar. Bir teklif talebinde aşağıdaki hususların belirtilmesi önemlidir.

- Ürün - Sistem Gereksinimleri: İşletmenizin ihtiyacı olan ürün veya sistemin ayrıntılı gereksinimlerini belirtmelisiniz. Bu gereksinimler, işlevsellik, performans, entegrasyonlar ve diğer teknik detayları içermelidir.

- Ürün Ölçeklenebilirliği ve Birlikte Çalışabilirlik: Ürününüzün gelecekte büyüme ve genişleme gereksinimlerinizi karşılayabilme yeteneği önemlidir. Ayrıca, diğer sistemlerle uyumlu çalışabilir olmalıdır.

- Müşteri Referansları: Tedarikçinin daha önce çalıştığı müşterilerin isimlerini ve iletişim bilgilerini talep ederek, tedarikçinin geçmiş performansını değerlendirebilirsiniz.

- Firma Yaşayabilirligi / Finansal İstikrar: Tedarikçinin mali durumunu ve finansal istikrarını değerlendirmek için gerekli finansal belgeleri ve bilgileri istemelisiniz. Bu, uzun vadeli bir iş ilişkisi için önemlidir.

- Tam ve Güvenilir Dokümantasyon Varlığı: Ürün veya sistemle ilgili tam, güncel ve anlaşılır dokümantasyonun mevcut olması gereklidir. Bu, ürünü daha iyi anlamanız ve sorunları çözmeniz için önemlidir.

- Firma Desteği: Tedarikçinin sunduğu müşteri desteği hakkında bilgi almalısınız. Sorunların nasıl çözüleceği ve destek süreçleri hakkında net bir anlayışa sahip olmalısınız.

- Kaynak Kod Mevcudiyeti: Özellikle özel yazılım geliştirme projeleri için, ürünün veya yazılımın kaynak kodunun sizin tarafınızdan erişilebilir ve denetlenebilir olması önemlidir.

- Teklif Edilen Ürün Üzerine Yıl Deneyimi: Tedarikçinin teklif ettiği ürün veya hizmetle ilgili deneyimini değerlendirmelisiniz. Daha önce benzer projelerde ne kadar süre ve deneyimleri olduğunu öğrenmelisiniz.

- Tarihleri ile Birlikte Yeni ve Planlanmış İyileştirme Listesi: Tedarikçinin ürün veya hizmetini nasıl geliştirmeyi planladığını ve gelecekteki güncellemelerin ne zaman yayınlanacağını anlamak için bu bilgilere ihtiyacınız vardır.

- Mevcut Kullanıcı Listesi ve Ürünü Kullanan Müşteri Sayısı: Ürünün veya hizmetin mevcut müşteri tabanını ve kullanıcılarını öğrenmek, ürünün popülerliği ve güvenilirliği hakkında bilgi sağlayabilir.

- Ürünün Kabul Testi: Ürünün veya sistemin nasıl test edileceğini ve kabul edileceğini belirtmelisiniz. Bu, ürünün istediğiniz gibi çalıştığını doğrulamak için önemlidir.

Yazılım Seçiminde Başlıca Teknolojik Yönler ve Riskler

Yazılım seçimi, yalnızca işlevsellik, maliyet ve kullanıcı ihtiyaçları açısından değil, aynı zamanda teknolojik uyum, güvenlik ve sürdürülebilirlik gibi kritik faktörler açısından da titizlikle yapılması gereken bir süreçtir. Bu süreçte dikkat edilmesi gereken bazı teknolojik yönler ve bunlarla ilişkili riskler aşağıda sıralanmıştır:

1. Bulut Teknolojileri (Cloud Technologies)

Bulut tabanlı çözümler, şirketlerin daha hızlı ve esnek bir şekilde dijital dönüşüm süreçlerini hayata geçirmelerini sağlar.

Ölçeklenebilirlik (Scalability): Bulut çözümleri, işletmelerin ihtiyacı arttıkça hızla büyüyebilmesini sağlar.

Esneklik (Flexibility): Farklı iş yüklerini yönetmek ve coğrafi olarak dağılmış kullanıcılar için verimli çözümler sunmak mümkündür.

Maliyet Avantajları (Cost Efficiency): Bulut teknolojileri, donanım yatırımları yerine abonelik bazlı ödeme modeli sunduğu için başlangıç maliyetlerini düşürür.

Riskler:

Veri güvenliği (Data Security): Bulut ortamlarında veri güvenliği ve gizliliği konuları oldukça kritik olabilir. Şirketlerin hassas verilerini üçüncü taraf sağlayıcılara emanet etmek, olası veri ihlalleri riskini artırabilir.

Hizmet kesintileri (Service Downtime): Bulut hizmet sağlayıcılarının yaşadığı kesintiler, tüm işletmenin iş süreçlerini olumsuz etkileyebilir.

2. Açık Kaynak Yazılımlar (Open Source Software)

Açık kaynak yazılımlar, yazılımın kaynak kodlarına erişim sağlayarak esneklik ve özelleştirme olanağı sunar.

Maliyet Etkinliği (Cost-Effectiveness): Açık kaynak yazılımlar genellikle ücretsiz veya düşük maliyetlidir.

Topluluk Desteği (Community Support): Büyük açık kaynak yazılım toplulukları, hızlı çözüm geliştirme ve hata ayıklama konusunda önemli bir kaynak sunar.

Riskler:

Destek Eksikliği (Lack of Support): Ticari yazılımlar gibi resmi destek almayı zorlaştırabilir. Bazı yazılımlar için, hata düzeltme veya yazılımın güncellenmesi süreci zaman alabilir.

Güvenlik Açıkları (Security Vulnerabilities): Açık kaynak yazılımlar genellikle kamuya açık olduğu için, kötü niyetli kişiler bu yazılımlardaki güvenlik açıklarını hedef alabilir.

Yazılım Uyumsuzluğu (Software Incompatibility): Diğer yazılımlar ve sistemlerle uyumsuzluklar, entegrasyon sorunlarına yol açabilir.

3. Yazılımın Performansı ve Hızlı Yanıt Süreleri (Software Performance & Response Times)

Yanıt Süresi (Response Time): Yazılımın kullanıcı taleplerine verdiği yanıt süresi, sistem performansının önemli bir göstergesidir. Hızlı yanıt süreleri, kullanıcı memnuniyetini artırır ve iş süreçlerinin verimliliğini destekler.

Yük Dengeleme (Load Balancing): Yazılımın eş zamanlı çoklu işlemleri yönetebilmesi önemlidir. Özellikle büyük ölçekli işletmelerde, yazılımın yüksek yük altında dahi stabil çalışması gerekir.

Riskler:

Sistem Çökmesi (System Crashes): Yüksek işlem hacmi veya kötü tasarlanmış yazılım, sistemin çökmesine neden olabilir. Bu durum, iş kaybına ve kullanıcı memnuniyetsizliğine yol açabilir.

Performansın Düşmesi (Performance Degradation): Sistemin zamanla performans kaybı yaşaması, yazılımın güncellenmesini veya iyileştirilmesini gerektirebilir.

4. Güvenlik ve Veri Koruma (Security & Data Protection)

Veri Şifreleme (Data Encryption): Verilerin güvenliğini sağlamak için şifreleme kullanmak, dışarıdan gelebilecek saldırılara karşı önemli bir koruma sağlar.

Kimlik Doğrulama ve Erişim Kontrolleri (Authentication & Access Control): Kullanıcıların sisteme erişim yetkilerinin yönetilmesi, hassas verilere sadece yetkilendirilmiş kişilerin ulaşmasını sağlar.

Riskler:

Veri Sızıntısı (Data Breaches): Verilerin üçüncü taraflarca ele geçirilmesi, hem finansal kayıplara hem de itibar kaybına yol açabilir.

Kimlik Avı Saldırıları (Phishing Attacks): Kullanıcıların kimlik bilgilerini çalan sahte yazılım ve uygulamalar, ciddi güvenlik ihlallerine neden olabilir.

5. Entegrasyon ve Uyumluluk (Integration & Compatibility)

Mevcut Sistemlerle Entegrasyon (Integration with Existing Systems): Yeni yazılımın mevcut IT altyapısına entegrasyonu, iş süreçlerinin kesintisiz devam etmesi için önemlidir.

Veri Uyumluluğu (Data Compatibility): Yeni yazılımın veri formatlarının ve yapılarının mevcut sistemle uyumlu olması, veri aktarımını kolaylaştırır.

Riskler:

Entegrasyon Hataları (Integration Errors): Farklı sistemler arasında veri paylaşımı veya entegrasyon sırasında uyumsuzluklar ve hatalar yaşanabilir.

Yazılım Güncellemeleri (Software Updates): Yeni yazılımlar entegre edildikçe, mevcut sistemlerin güncellenmesi gerekebilir ve bu da ilave maliyetlere yol açabilir.

6. Yazılımın Ölçeklenebilirliği (Software Scalability)

Yazılımın ölçeklenebilirliği, işletmenin büyümesiyle uyumlu olarak yazılımın da büyümesini ve gelişmesini sağlar. Yüksek kullanıcı sayıları, artan veri işleme talepleri ve yeni iş alanlarına uyum sağlayabilmek için yazılımın uygun ölçeklenebilirlik özelliği olması gerekmektedir.

Riskler:

Yetersiz Ölçeklenebilirlik (Insufficient Scalability): Yazılım, şirket büyüdükçe sorunlara yol açabilir, bu da yazılımın değiştirilmesini veya yeniden tasarlanmasını gerektirebilir.

7. Yazılımın Güncellemeleri ve Bakımı (Software Updates & Maintenance)

Yazılımın düzenli olarak güncellenmesi ve bakımı, yeni özelliklerin eklenmesi ve eski hataların giderilmesi için gereklidir. Yazılımın güncel tutulması, güvenlik açıklarının kapatılması ve sistem performansının iyileştirilmesi için kritik öneme sahiptir.

Riskler:

Güncelleme Zamanlamaları (Update Schedules): Yazılım güncellemelerinin zamanlaması, iş süreçlerini etkileyebilir. Duygusal hatalar ve kesintiler nedeniyle güncellemeler sırasında oluşabilecek olumsuz durumlar, operasyonel aksamalara yol açabilir.

Aşama 3B- Tasarım (İşletme İçi Geliştirme):

İşletme içinde yapılacak geliştirmeler için tasarım süreci gerçekleştirilir. Bu süreçte tamamlanmış olan gereksinimlere dayanarak, işletme içi geliştirmelerin gereği olarak sistem bölümlerini arayüzlerini ve seçilen donanım, yazılım ve ağ olanaklarını kullanarak sistemin nasıl uygulanacağını

tanımlayan bir sistem ve alt sistem spesifikasyonları için bir taban çizgisi oluşturulur. Tasarım genellikle program ve veri tabanı özelliklerini de içerir ve gerekli tüm güvenlik hususlarını ele alır. Bununla beraber, yeni gereksinimlerin kontrolsüz bir şekilde geliştirme sürecine girmesini engellemek için değişim kontrol süreci oluşturulur.

Tasarım sürecinin temel unsurları:

Gereksinimlere Dayalı Tasarım: İşletme içi geliştirmelerin tasarımı, öncelikle tamamlanmış gereksinimlere dayanır. Bu gereksinimler, işletme içindeki ihtiyaçları ve hedefleri belirler.

Sistem ve Alt Sistem Spesifikasyonları: Tasarım süreci, işletme içi geliştirmelerin nasıl uygulanacağını ayrıntılı bir şekilde tanımlar. Bu, sistem ve alt sistemlerin nasıl çalışacağını, arayüzlerin nasıl olacağını ve hangi donanım, yazılım ve ağ kaynaklarının kullanılacağını içerir.

Program ve Veritabanı Özellikleri: Tasarım aşamasında, işletme içi geliştirmeler için kullanılacak yazılım programları ve veritabanı özellikleri belirlenir. Bu, uygulamanın işlevselliği ve veri yönetimi için kritik öneme sahiptir.

Güvenlik Hususları: Tasarım süreci, işletme içi geliştirmelerin güvenliği için gerekli tüm önlemleri ele alır. Veri güvenliği, erişim kontrolü ve diğer güvenlik önlemleri tasarımın ayrılmaz bir parçasıdır.

Değişim Kontrol Süreci: Tasarım aşamasında, yeni gereksinimlerin veya değişikliklerin kontrollü bir şekilde ele alınması için bir değişim kontrol süreci oluşturulur. Bu süreç, kontrolsüz değişikliklerin projeyi etkilemesini engeller.

Bu aşamada yazılım geliştirme ile ilişkili risklerinde göz önüne alınması gerekir. Bu riskler başlıca aşağıdaki kategorilerde değerlendirilir.

Stratejik Risk

Kurumsal stratejiler dikkate alınmadan iş amaçları tanımlandığında ve önceliklendirildiği zaman ortaya çıkar.

İşletme içi yazılım geliştirme sürecinde karşılaşılabilecek stratejik risklere örnekler:

- Pazar Değişiklikleri: Pazarın hızla değiştiği bir sektörde işletme, yazılım geliştirme projelerini başlatırken pazarın değişikliklerini göz ardı edebilir. Bu, yazılımın pazara uyumlu olmamasına ve rekabet avantajını kaybetmesine yol açabilir.

- Teknolojik Gelişmeler: Hızla gelişen teknoloji, yazılım geliştirme projelerinin uzun vadeli teknolojik uygunluğunu tehdit edebilir. İşletme, teknolojik gelişmeleri dikkate almadan projelere başlarsa, yazılımın hızla eskimesine neden olabilir.

- Finansal Riskler: Yazılım projeleri maliyetli olabilir ve bütçe aşımı veya finansal zorluklar işletme için stratejik bir risk oluşturabilir.

- Rekabet Baskısı: Rekabetin yoğun olduğu sektörlerde, işletme rekabet avantajını kaybetme riskiyle karşı karşıya olabilir. Yazılım projelerinin rekabet avantajını sürdürebilmesi için stratejik düşünülmesi gerekebilir.

- Yönetişim Sorunları: Yazılım projelerinin yönetişimi ve koordinasyonu zor olabilir. Eğer bu konularda eksiklikler varsa, projeler gecikebilir veya istenilen sonuçları üretemeyebilir.

- Tedarikçi Sorunları: Yazılım geliştirme projeleri sıkça dış tedarikçilerle çalışmayı gerektirir. Yanlış tedarikçi seçimi veya tedarikçi ilişkilerinin yönetiminde sorunlar, projelerin başarısızlığına yol açabilir.

- Yönetim ve Liderlik Eksikliği: Yazılım projelerinin etkili bir şekilde yönetilmemesi veya liderlik eksikliği, projelerin stratejik hedeflere uygun olarak ilerlememesine neden olabilir.

İş Riski

Oluşturulan sistemin kullanıcı ihtiyaçlarına cevap verememesi, gereksinimlerini karşılayamaması riski.

İş riski, yazılım geliştirme projelerinde sıkça karşılaşılan bir stratejik risk türüdür. Bu risk, oluşturulan sistemin kullanıcı ihtiyaçlarına veya işletme gereksinimlerine cevap verememesi veya bu gereksinimleri karşılayamaması durumunda ortaya çıkar.

İşletme içi yazılım geliştirme sürecinde karşılaşılabilecek iş risklerine örnekler:

- Gereksinimlerin Yanlış Anlaşılması: Yazılım projesinin başlangıcında gereksinimlerin eksik veya yanlış anlaşılması, son ürünün kullanıcı ihtiyaçlarını karşılayamamasına neden olabilir.

- Değişen İş İhtiyaçları: İşletme ihtiyaçları zaman içinde değişebilir ve yazılım projesi bu değişiklikleri takip etmezse, kullanıcı ihtiyaçlarına cevap veremeyebilir.

- Kapsam Kontrolü: Projenin kapsamının kontrolsüz bir şekilde büyümesi, projenin zamanında tamamlanmasını zorlaştırabilir ve bu da kullanıcı ihtiyaçlarına uygun bir çözüm sunma yeteneğini azaltabilir.

- İş Süreçlerinin Yanlış Otomasyonu: İş süreçlerinin yanlış şekilde otomasyonu veya gereksinimlerin eksik anlaşılması, işletmenin iş süreçlerini düzgün bir şekilde desteklemeyen bir yazılım sonucu doğurabilir.

- Teknik Sorunlar: Yazılım geliştirme sürecinde teknik sorunlar, sistemin stabilitesini veya performansını etkileyebilir ve bu da kullanıcı ihtiyaçlarına uygun bir hizmet sunma yeteneğini düşürebilir.

İş riskleri, yazılım geliştirme projelerinde çok önemli bir stratejik risk türüdür çünkü işletmenin başarısı için önemli olan kullanıcı ihtiyaçlarına uygun bir yazılım çözümünün oluşturulamaması durumunda, iş stratejileri ve rekabet avantajı tehlikede olabilir. Bu nedenle, gereksinimlerin doğru anlaşılması, değişen iş ihtiyaçlarına uyum sağlama yeteneği ve projenin kapsamının kontrol altında tutulması gibi faktörlere dikkat edilmesi önemlidir.

Proje Riski

Proje için planlanan finansal kaynakların planlananı aşması sonucu projenin öngörülen zaman dilimi içinde tamamlanamama riski.

Bilgi sistemleri denetçisi, sadece bir SDLC yaklaşımı izlemenin, bir geliştirme projesinin başarılı bir şekilde tamamlanmasını sağlamadığını bilmelidir. Bilgi sistemleri denetimi kapsamında aşağıdaki hususlarında değerlendirilmesi gerekmektedir.

- Yönetim tarafından yazılım tasarım ve geliştirme faaliyetlerinin takip edildiği ve bu faaliyetlere destek verildiği,

- Proje aşamalarında periyodik gözden geçirildiği ve risk analizinin yapıldığı,

- Projenin amaç ve hedefi sağlayıp sağlamadığı,

- Kaynak bütçe ve zamanlama planlamasının doğru bir şekilde yapıldığı,

- Planlamalardan sapmaların kontrol edilip edilmediği imkânsız sapmaların önüne geçmek amacıyla yazılım için bir taban çizgisinin oluşturulup oluşturulmadığı.

Yapısal analiz, tasarım ve geliştirme tekniklerinin kullanımı klasik bir SDLC için vazgeçilmezdir. Bilgi sistemleri denetçisi de özellikle geleneksel bir SDLC yaklaşımı kullanırken detaylı bir şekilde tanımlandığını, belgelendiğini ve takip edildiğini değerlendirmelidir.

Ayrıca genel bir ön tasarımın oluşturulmasında varlık ilişkisi diyagramlarının kullanılması son derece önemlidir.

Yazılım tasarım aşamasında yazılım taban çizgisi (tasarım dondurma) de oluşturulur. Yazılım taban çizgisi tasarımdaki kesme noktası anlamına gelir. Kullanıcı gereksinimleri zaman, fayda etki ve maliyet açısından değerlendirilerek belirlenmiş noktadır. Sistem gereksinimlerini taban çizgisi üzerinden yönetememek proje kapsamında risklerin gerçekleşmesine neden olur.

Tasarımın başarılı olması için kısıtlı da olsa kullanıcıların katılımı önemlidir. Katılım kısıtlıdır, çünkü kullanıcılardan teknik detayda bilgi sahibi olmaları beklenmez. Bu noktada bazı önemli noktalara aşağıda yer verilmektedir.

- Kullanıcı Gereksinimleri Odaklı: Yazılım taban çizgisi, kullanıcı gereksinimlerini temel alır. Kullanıcıların beklentileri ve ihtiyaçları göz önünde bulundurularak tasarımın belirli bir aşamasında bu gereksinimlere ulaşılmasının beklenir.

- Risk Yönetimi: Yazılım taban çizgisi, projenin ilerleyişini kontrol etmek ve riskleri minimize etmek için önemlidir. Her değişiklik veya eklenen özellik tasarımı karmaşıklaştırabilir ve maliyeti artırabilir.

- Kullanıcı Katılımı: Tasarım sürecinin başarılı olması için kullanıcıların katılımı önemlidir. Ancak, kullanıcıların teknik detaylara hakim olmaları beklenmez. Onların geri bildirimleri ve gereksinimleri, tasarım sürecinin doğru yönlendirilmesine yardımcı olabilir.

- Dengeli Yaklaşım: Yazılım taban çizgisi belirlenirken, kullanıcı gereksinimlerini karşılamak için aynı zamanda projenin zaman çizelgesini ve maliyetini dengeli bir şekilde yönetmek önemlidir.

- İlerleme İzleme: Yazılım taban çizgisi belirlendikten sonra, projenin ilerlemesi düzenli olarak izlenmelidir. Herhangi bir değişiklik veya sapma, dikkatle değerlendirilmeli ve gerektiğinde kontrol altına alınmalıdır.

Tasarımda bilgi sistemleri denetçisi rolü aşağıdaki gibi özetlenebilir:

- Kalite Güvence Sonuçlarının Değerlendirilmesi: Bu görev, tasarım sürecinin kalite güvence standartlarına uygunluğunu değerlendirmeyi içerir. Bu, tasarımın işletme gereksinimlerini ve standartları karşılayıp karşılamadığını kontrol etmek için yapılan bir inceleme sürecini içerir.

- Risk Değerlendirmelerinin Tamlığı, Bütünlüğü ve Doğruluğuna Güvence Verilmesi: Bu görev, tasarım sürecinin risk yönetimi açısından eksiksiz ve doğru bir şekilde yapıldığını doğrulamayı içerir. Risk değerlendirmelerinin tam ve güvenilir olduğundan emin olmak, projenin gelecekteki risklere karşı hazırlıklı olmasını sağlar.

- Sistem Girdi ve Çıktılarının İncelenmesi: Bu görev, tasarımın sistem girdi ve çıktılarını doğru ve eksiksiz bir şekilde ele aldığını doğrulamayı içerir. Verilerin doğru bir şekilde alındığını ve işlendiğini kontrol etmek için yapılır.

- Sistem Akış Diyagramlarının Değerlendirilmesi: Bu görev, tasarımın iş akışını ve süreçlerini anlamak için kullanılan akış diyagramlarını değerlendirmeyi içerir. Bu, süreçlerin mantıklı ve etkili bir şekilde tasarlandığını doğrulamayı amaçlar.

- Denetim İzlerinin Yeterliliğine Güvence Verilmesi: Bu görev, denetim izlerinin tasarımın gereksinimlerini ve standartlarını karşıladığını doğrulamayı içerir. Denetim izlerinin yeterliliği, tasarımın izlenebilir ve denetlenebilir olduğundan emin olmak için önemlidir.

- Hesaplamaların ve İşlemlerin Bütünlüğe Güvence Verilmesi: Bu görev, tasarımın hesaplamaların ve işlemlerin doğruluğunu ve bütünlüğünü sağlayacak şekilde tasarlandığını doğrulamayı içerir. Hesaplamaların ve işlemlerin hatasız ve güvenilir olduğundan emin olmak önemlidir.

- Hatalı Verilerin Sistem Tarafından Doğru ve Eksiksiz Bir Şekilde İşlenmesine Güvence Verilmesi: Bu görev, tasarımın hatalı verileri algılayabilme ve işleyebilme yeteneğini doğrulamayı içerir. Bu, sistemin veri doğruluğunu ve güvenilirliğini sağlamaya yardımcı olur.

Aşama 4A- Yapılandırma, Konfigüre Etme (Satın Alınan Sistemler):

Bir paket program sisteminin temin edildikten sonra işletmenin gereksinimlerine göre uyarlaması için yapılandırılması gerekmektedir. Bu amaçla genel olarak programlarda yazılımsal değişiklikler yapmak yerine genel olarak sistem kontrol parametreleri kullanılarak sistemin ihtiyaca uygun şekilde kontrol edilmesi sağlanır. Günümüzün yazılım paketleri son derece esnek oldukları için paketin işlevselliğini ilgili parametreleri açıp kapatarak veya tablolardaki parametreleri ayarlayarak işletmeye adapte edilmesi sağlanır. Eğer sadece parametre ayarları yeterli olmuyorsa bu aşamada ayrıca

satın alınan sistemin mevcut programlara ve veri tabanlarına bağlanacak arayüz programlarının da geliştirmesi gerekebilir. Bu aşamada iş surecilerinde istenmeyen sonuçları en aza indirmek için BT sistemlerindeki değişiklikler dikkatlice değerlendirilmeli, planlanmalı, test edilmeli, onaylanmalı, belgelenmeli ve anlatılmalıdır.

Bilgi sistemleri denetçisinin, geliştirme personeli ile üretim ortamı arasında görevler ayrılığını (SoD) sağlamak için konfigürasyon, değişim ve sürüm yönetimi ve mevcut kontrollerin yönetimi için mevcut araçların farkında olması gerekir.

Konfigürasyon yönetim araçları, aşağıdakileri kullanarak değişiklik yönetimini ve sürüm yönetimini destekler:

- Fonksiyonel, operasyonel ve güvenlik açısından etki değerlendirmesi yapılabilmesini teminen önerilen bir değişiklikten etkilenen ögelerin belirlenmesi,

- Değişikliklerin yetkilendirme kayıtlarına uygun olarak uygulanması,

- İnsan hatalarından ve kaynak maliyetlerinden kaçınmak için genel bir sürüm hazırlanması,

- Yetkili değişikliklerden etkilenen yapılandırma ögelerinin doğru olarak belirlenip, sürümler devreye alındığında yapılandırma öğesindeki tüm değişikliklerinin kaydedilmesi,

- Uygulanan bir değişiklik başarısız olursa bir işletmenin geri döneceği sürümlerle ilgili (öncesinde doğru olarak çalıştığı bilinen) referans noktasının kaydedilmesi.

Konfigürasyon, konfigüre etme ve yazılım paketlerinin işletmeye uyarlama, bilgi teknolojileri (BT) ve bilgi sistemleri yönetimi açısından kritik bir rol oynar. Aşağıda bu konular hakkında daha fazla açıklama bulabilirsiniz:

- Konfigürasyon Yönetimi Nedir?: Konfigürasyon yönetimi, bir yazılım paketinin veya sistemini, belirli işletme gereksinimlerini karşılamak için ayarlama ve yapılandırma sürecini ifade eder. Bu süreç, genellikle yazılımsal değişiklikler yapmadan, sistemin parametrelerini veya ayarlarını değiştirerek gerçekleştirilir. Amaç, yazılımın veya sistemin işletme ihtiyaçlarına tam olarak uymasını sağlamaktır.

- Yazılım Paketi Uyarlama: Günümüzün yazılım paketleri son derece esnek olup, birçok işletmenin gereksinimlerine uygun hale getirilebilirler. Bu uyarlama, genellikle sistem kontrol parametrelerinin kullanılmasıyla yapılır. Bu, işletmenin belirli iş süreçlerini veya gereksinimlerini karşılamak için yazılımın farklı özelliklerini açma, kapatma veya ayarlama anlamına gelir.

- Arayüz Geliştirme: Eğer yazılım paketi mevcut programlara veya veri tabanlarına bağlanacaksa, özelleştirilmiş arayüz programlarının geliştirilmesi gerekebilir. Bu arayüzler, farklı sistemler arasındaki veri akışını ve etkileşimini kolaylaştırır. Bu aşamada, iş süreçlerinin kesintiye uğramadan veri entegrasyonunu nasıl yöneteceğiniz önemlidir.

- Konfigürasyon Yönetimi Araçları: Konfigürasyon yönetimi ve değişiklik yönetimi süreçlerini desteklemek için özel araçlar ve yazılımlar kullanılır. Bu araçlar, değişikliklerin kaydedilmesi, yetkilendirilmesi, geri alınması ve izlenmesi gibi işlemleri kolaylaştırır.

- Risk Yönetimi: Konfigürasyon değişiklikleri ve yazılım uyarlama işlemleri iş süreçlerinde potansiyel riskler oluşturabilir. Bu nedenle, bu tür değişikliklerin önceliklendirilmesi, dikkatlice yönetilmesi ve test edilmesi gerekmektedir. İş süreçlerinde istenmeyen sonuçları en aza indirmek için risk değerlendirmesi yapılmalıdır.

- Dokümantasyon ve İletişim: Konfigürasyon yönetimi süreçleri sırasında yapılan değişiklikler ve ayarlamalar dikkatlice belgelenmelidir. Ayrıca, bu değişikliklerin ilgili paydaşlara (örneğin, denetçiler ve geliştirme personeli) iletilmesi ve onaylanması önemlidir.

Konfigürasyon yönetimi, bir organizasyonun yazılım ve sistemlerini düzgün bir şekilde yönetmesine ve iş süreçlerinin gereksinimlerine uygun hale getirmesine yardımcı olan önemli bir süreçtir. Bu süreç, iş süreçlerinin düzgün çalışmasını sağlamak ve BT sistemlerinin güvenilirliğini artırmak için dikkatle yönetilmelidir.

Aşama 4B- Geliştirme (İşletme İçi Geliştirme):

Geliştirme, proje sahibinin talepleri doğrusunda yapılmış olan tasarıma uygun yazılım geliştirme araçlarıyla vasıtasıyla ürüne dönüştürme işlemidir. Bu aşamada veri tabanının oluşturulması, kodlamanın yapılması, kullanıcı ara yüzlerinin oluşturulması ve raporların yazılması gibi işlemler yapılır. Projeye ait ilk çıktılar ortaya çıkmaya başlar. Geliştirilen modüllerle ilgili birim testleri ve sistem testleri de bu aşamada yapılabilir.

Bu süreçte yazılımın güvenliği, işlevselliği ve sürdürülebilirliği için önemli bir dizi faktör göz önünde bulundurulmalıdır. Yazılım geliştirme sürecinin başlangıcından itibaren güvenlik, kalite ve izlenebilirlik ön planda tutulmalıdır. Yazılım geliştirme sürecinde, yazılımın güvenliği, veritabanı işlemleri ve kullanıcı doğrulama süreçlerine özel güvenlik önlemleri eklenmelidir. Veritabanı işlemleri sırasında SQL enjeksiyonu gibi saldırılara karşı koruma sağlayan güvenlik önlemleri gereklidir. Ayrıca, kullanıcıların kimlik doğrulama süreçlerinde güçlü şifreleme teknikleri kullanılmalı ve verilerin gizliliği korunmalıdır. Bu güvenlik önlemleri yazılımın tasarım aşamasından itibaren uygulanmalı ve her geliştirme aşamasında gözden geçirilmelidir. Ek olarak, yazılım geliştirme sürecinin şeffaflığı ve izlenebilirliği de büyük önem taşır. Yazılım geliştirme süreçlerinde şeffaflık, tüm geliştirme ve test aşamalarının izlenebilir olması gerektiği anlamına gelir. Sürüm kontrol sistemleri, kodun her aşamasının izlenmesini ve geliştirilen yazılımın her bir değişikliğini kaydetmeye olanak tanır. Bu sistemler, yazılımın önceki sürümleriyle uyumluluğunu sağlamada ve oluşabilecek hataları hızlıca tespit etmede kritik bir rol oynar. Ayrıca, yazılım geliştirme sürecindeki her değişiklik, denetçiler ve geliştirme personeli tarafından onaylanmalı ve ilgili paydaşlarla düzenli olarak paylaşılmalıdır.

Programda daha kaliteli bir sonuç elde etmek ve bakımların etkin ve efektif bir şekilde yapılabilmesini teminen kodlama standartları uygulanmaktadır. Bu durum kodların daha basit yazılmasını okunmasını ve anlanmasına imkân vermektedir.

Programların daha kolay yazılabilmesini teminen sadece programlama standartları değil çevirim için programlama olanakları da değerlendirilmektedir. Çevirim içi programlama olanakları bilgisayar iş istasyonlarında kullanılır. Çevrim içi sistemler programcının sorun çözme yeteneğini geliştirir ama yetkisiz erişimlerden oluşan güvenlik açıkları oluşturur.

Sistem geliştirme işlemi sırasında birçok programlama hatası tespit edilebilir. Bu kapsamda hata ayıklama araçları kullanılır. Hata ayıklama araçları üç ana kategoriye ayrılır. Bunlar:

Mantık Yolu Monitörleri:

Mantık yolu monitörleri, yazılımın mantıksal akışını izlemek ve kontrol etmek için kullanılır.

Programın her adımını takip eder ve beklenmeyen veya istenmeyen durumları tespit eder.

Kodun hangi yollarının izlendiğini ve hangi yolların izlenmediğini gösterir.

Kodun akışını görsel olarak analiz etme imkanı sunar ve böylece mantıksal hataları tespit etmeye yardımcı olur.

Bellek Dökümleri:

Bellek dökümleri, programın bellek kullanımını izlemek ve bellek sızıntıları gibi problemleri tespit etmek için kullanılır.

Programın çalışma sırasında hangi bellek alanlarını kullandığını ve ne kadar bellek tükettiğini gösterir.

Bellek sızıntılarını tespit eder, yetersiz bellek tahsisi sorunlarını belirler ve aşırı bellek kullanımını izler.

Programın performansını artırmak ve istenmeyen bellek sorunlarını gidermek için önemlidir.

Çıktı Analizörleri:

Çıktı analizörleri, programın çıktılarını izlemek ve analiz etmek için kullanılır.

Programın ürettiği verileri kontrol eder ve beklenmeyen sonuçları tespit eder.

Test senaryolarının sonuçlarını karşılaştırarak hataları ve tutarsızlıkları ortaya çıkarır.

Özellikle karmaşık sistemlerde ve büyük veri işleme uygulamalarında kullanışlıdır.

Aşama 5- Kullanıcı Kabul Testi ve Uygulamaya Alma:

Bu aşamada gerçekleştirilen kullanıcı kabul testinin son tekrarı ve takibinde kullanıcıdan alınan onay ile yeni bilgi sisteminin fiili olarak devreye alınarak çalıştırılması sağlanır. Bu süreçte riskleri kabul edilebilir bir seviyeye indirgemesi ve sistemin etkinliği üzerinde yönetimin sorumluluğunun sağlanması, sistemin amaçlanan hedeflerini karşılaması ve uygun bir iç kontrol seviyesinin oluşturulması konusunda iş uygulamasının etkinliğinin değerlendirilmesi için bir spesifikasyon ve akreditasyon sürecinden de geçebilmektedir.

Kullanıcı kabul testinin başarısı, test senaryolarının kapsamlı ve iyi yapılandırılmış olmasına bağlıdır. Bu testlerin, yazılımın tüm fonksiyonel gereksinimlerini kapsayacak şekilde belirlenmesi gerekir. Ayrıca, test sırasında oluşan her hata ya da eksiklik, yazılımın fiili olarak devreye alınmadan önce düzeltilmeli, bu da son kullanıcı deneyiminin iyileştirilmesine katkı sağlar.

Kullanıcı kabul testi ve uygulamaya alma süreci aşağıdaki temel bileşenleri içerir:

- Kullanıcı Kabul Testi: Yeni bir bilgi sistemi veya yazılım uygulaması kullanıcılar tarafından test edilir. Kullanıcılar, sistemin tasarımına, işlevselliğine ve kullanılabilirliğine göre değerlendirme yaparlar. Bu test, yazılımın kullanıcı ihtiyaçlarına uygun olup olmadığını belirlemeye yardımcı olur.

Kullanıcı Kabul Testi Örneği: Bir şirket, yeni bir müşteri ilişkileri yönetimi (CRM) yazılımı uygulamaya almayı planlıyor. Kullanıcı kabul testi süreci sırasında, şirketin satış ekibi bu yeni CRM yazılımını kullanarak gerçek müşteri verileriyle çalışır. Kullanıcılar, yazılımın kullanıcı dostu olup olmadığını, müşteri bilgilerini doğru bir şekilde kaydedip kaydetmediğini ve satış işlemlerini daha verimli hale getirip getirmediğini değerlendirirler. Test sonuçlarına dayanarak yazılımın kullanıma alınıp alınmayacağına karar verilir.

- Onay ve Devreye Alma: Kullanıcıların test sonuçlarına dayanarak yazılımı onaylaması ve sistemi devreye alması gerekebilir. Bu aşamada, yazılımın canlı ortamda kullanılmaya başlaması sağlanır.

Onay ve Devreye Alma Örneği: Bir e-ticaret platformu, yeni bir ödeme işleme sistemi entegre etmeyi planlıyor. Testler tamamlandığında, platformun yönetimi yeni ödeme işleme sisteminin güvenilir olduğunu ve işlem sırasında sorun çıkartmadığını doğrular. Bu onayın ardından, yeni ödeme işleme sistemi canlı ortamda kullanılmaya başlanır.

- Risk Yönetimi: Bu süreç, kullanıma alma sırasında ortaya çıkabilecek riskleri tanımlamayı, analiz etmeyi ve kabul edilebilir seviyelere indirgemeyi içerir. Riskler, iş sürekliliği, veri kaybı veya güvenlik ihlali gibi çeşitli alanlarda olabilir.

Risk Yönetimi Örneği: Bir banka, online bankacılık uygulamasının güvenliğini artırmak için yeni bir güvenlik protokolü uygulamaya alır. Yeni protokolün potansiyel riskleri değerlendirilir ve gerekli önlemler alınır. Örneğin, iki faktörlü kimlik doğrulama (2FA) kullanarak müşteri hesaplarının güvenliğini artırmak, potansiyel kimlik hırsızlığı riskini azaltabilir.

- İç Kontrol: Uygulamaya alma aşamasında iç kontrol önemlidir. İç kontrol, iş süreçlerinin doğru çalışmasını, güvenliği ve veri bütünlüğünü sağlama amacını taşır.

İç Kontrol Örneği: Bir sağlık kuruluşu, tıbbi kayıtları dijital olarak yöneten bir sistem kullanmaktadır. Bu sistemde, tıbbi bilgilerin gizliliği ve veri bütünlüğü son derece önemlidir. İç kontrol önlemleri, sadece yetkilendirilmiş personelin erişimine izin verir ve veri kaybını veya veri bozulmasını önlemek için düzenli yedeklemeler yapar.

- Spesifikasyon ve Akreditasyon: Bazı durumlarda, belirli bir bilgi sistemi veya yazılım uygulamasının belirli standartlara veya gereksinimlere uygun olduğunu belgelemek amacıyla spesifikasyon ve akreditasyon süreçleri geçilebilir. Bu, özellikle hassas veri işleyen veya düzenlemelere tabi olan alanlarda önemlidir.

Spesifikasyon ve Akreditasyon Örneği: Bir hükümet kurumu, özellikle hassas bilgileri işleyen bir bilgi sistemi için yüksek güvenlik gereksinimlerine sahip bir yazılım kullanır. Bu yazılımın, belirli bir güvenlik standardına (örneğin, ISO 27001) uygun olduğunu doğrulamak için bir akreditasyon sürecinden geçmesi gerekebilir. Bu süreç, yazılımın güvenliğini ve uyumluluğunu belgeleyerek güvenilirliğini artırır.

Aşama 6- Uygulama Sonrası İnceleme:

Bakım sureci, yazılım kullanılmaya başlandıktan sonra sistem üzerinde değişiklik yapılması surecidir. Yazılım proje sahibi ve paydaşları tarafından yapılan isteklere göre sürekli değişir. Bu aşamada sisteme yeni eklentiler eklenir ya da hatalar giderilir.

Teslimden sonra değişiklik yapmanın temelde 3 nedeni vardır. Bunlar:

1. Doğrulayıcı Bakımlar: Bu tür bakımlar, sistemin mevcut hatalarını gidermek için yapılır. Hatalar, gereksinimlerin yanlış anlaşılması, tasarım hataları, kodlama hataları veya dökümantasyon eksiklikleri gibi nedenlerle ortaya çıkabilir. Doğrulayıcı bakımlar, yazılımın istenilen işlevselliği sağlamasını ve düzgün çalışmasını temin etmek için yapılır. Bu aşamada hataları tespit etmek, düzeltmek ve yazılımı güncellemek önemlidir.

Bu bakım türü kapsamında aşağıdaki örnekler verilebilir:

Hata Düzeltmeleri: Bir kullanıcı, belirli bir işlemi gerçekleştirirken hata mesajı alıyorsa, bu hata düzeltme bakımının bir örneğidir. Örneğin, kullanıcı girişi hatalı bir şekilde doğrulanıyorsa veya veritabanına veri kaydederken bir hata oluşuyorsa, bu hatalar düzeltilmelidir.

Dökümantasyon Güncellemeleri: Yazılımın dökümantasyonunda eksik veya hatalı bilgiler bulunabilir. Doğrulayıcı bakım, bu belgelerin güncellenmesini ve düzeltilmesini içerir.

2. Kusursuzlaştırma Bakımları: Kusursuzlaştırma bakımları, yazılımın performansını, güvenilirliğini ve kullanılabilirliğini artırmak için yapılır. Bu değişiklikler, mevcut işlevselliği optimize etmeyi veya yeni özellikler eklemeyi amaçlar. Kusursuzlaştırma bakımları, yazılımın daha etkin ve verimli çalışmasını sağlar.

Bu bakım türü kapsamında aşağıdaki örnekler verilebilir:

Performans İyileştirmeleri: Bir uygulama yavaş çalışıyorsa, kodun optimize edilmesi veya veritabanı sorgularının hızlandırılması gibi performans iyileştirmeleri yapılabilir.

Kullanıcı Arayüzü Güncellemeleri: Kullanıcı deneyimini artırmak için, kullanıcı arayüzüne yeni özellikler veya kullanılabilirlik geliştirmeleri eklemek kusursuzlaştırma bakımının bir parçası olabilir.

3. Çevresel Değişikliklere Cevap Verme: Yazılım, işletim sistemi güncellemeleri, tarayıcı değişiklikleri, güvenlik güncellemeleri veya harici entegrasyonlar gibi çevresel değişikliklere uyum sağlamalıdır. Bu tür değişiklikler, yazılımın sorunsuz bir şekilde çalışmaya devam etmesi için yapılır.

Örneğin, bir tarayıcı güncellendiğinde, web tabanlı bir uygulamanın bu güncellemeye uygun hale getirilmesi gerekebilir. Başka bir örnek, bir güvenlik açığı tespit edildiğinde, yazılımın bu açığı kapatması gerekir. Yazılımın güncel sürümü, bilinen bir güvenlik açığını düzeltmelidir.

Yeni ve yaygın olarak geliştirilmiş bir sistemin başarılı bir şekilde oluşturulmasını takiben, fizibilite aşaması için öngörülen değerlerin doğrulanmasına geçilir. Buradaki sapmalar karşısında sistemin yeterliliğini ve öngörülen maliyet-fayda ve yatırım getirisi ölçümlerini değerlendiren resmi bir süreç uygulanmaktadır. Bu yolla, bilgi sistemleri projesi ve son kullanıcı yönetimi, sistem eksiklerini ele almak için öğrenilen dersleri ve gelecekteki projeler için sistem geliştirme ve proje yönetimi için önerileri sağlayabilir.

Bilgi sistemleri denetçisi SDLC sürecini gözden geçirirken, çeşitli aşamalardan belgeler/dökümanlar almalı ve proje geliştirme toplantılarına katılarak sistem geliştirme sürecini gözlemlemeledir. Bilgi sistemleri denetçisi, proje ekibinin söz verilen/taahhüt edilen tarihlere kadar anahtar çıktıları üretme yeteneğini de değerlendirmelidir.

Tipik olarak, bilgi sistemleri denetçisi, aşağıdaki proje yönetimi faaliyetlerinin yeterliliğini gözden geçirmelidir:

Proje Komitesi/Kurulu Gözetimi Seviyeleri: Projenin yönetim ve denetim yapısını incelemelidir. Proje komitesi veya kurulu, projenin stratejik hedeflerle uyumlu olduğundan ve kaynakların etkili bir şekilde kullanıldığından emin olmalıdır.

Proje Dahilindeki Risk Yönetimi Yöntemleri: Risk yönetimi planını ve uygulama sürecini incelemelidir. Projede potansiyel risklerin belirlenmesi, değerlendirilmesi ve uygun risk azaltma veya kabul stratejilerinin oluşturulması önemlidir.

Sorun Yönetimi: Sorunların kaydedilmesi, takip edilmesi, öncelendirilmesi ve çözüm bulunması için bir sistem veya süreç olup olmadığını kontrol etmelidir. Sorunlar zamanında ve etkili bir şekilde çözülmelidir.

Maliyet Yönetimi: Proje maliyet tahminleri ve gerçek maliyetlerin izlenmesi süreçlerini değerlendirmelidir. Bütçe aşımları veya maliyet sapmaları varsa, bunların nedenleri ve düzeltici önlemler incelenmelidir.

Planlama ve Bağımlılık Yönetimi Süreçleri: Proje planının ve bağımlılıkların yönetimini gözden geçirmelidir. Proje zaman çizelgesinin ve kaynak planlarının uygunluğunu değerlendirmelidir.

Üst Yönetime Raporlama Süreçleri: Projenin ilerlemesi ve performansı hakkında üst yönetim için düzenli raporların nasıl hazırlandığını ve sunulduğunu kontrol etmelidir.

Kontrol Süreçlerinin Değiştirilmesi: Kontrol süreçlerinin projenin gereksinimlerine ve değişen koşullara nasıl uyum sağladığını değerlendirmelidir. Değişiklik yönetimi süreçlerinin uygunluğu da önemlidir.

Paydaş Yönetimi Katılımı: Proje paydaşlarının tanımlandığını, ihtiyaçlarının anlaşıldığını ve iletişim stratejilerinin bu ihtiyaçları karşılayacak şekilde nasıl uygulandığını incelemelidir.

İmzalama Süreci: Projenin önemli aşamalarında veya maliyet aşımlarında üst yönetim veya ilgili paydaşlardan imzalı onayların nasıl alındığını değerlendirmelidir. Bu onaylar, projenin ilerlemesini ve mali durumunu denetlemek için kullanılır.Ek olarak, SDLC surecinin tüm aşamalarının yeterli ve eksiksiz bir şekilde dokümantasyonu olmalıdır.

Tipik dokümantasyon türleri, bunlarla sınırlı olmamak üzere aşağıdakileri içerir:

- Her aşamada neyin gerçekleştirileceğini tanımlayan hedefler,

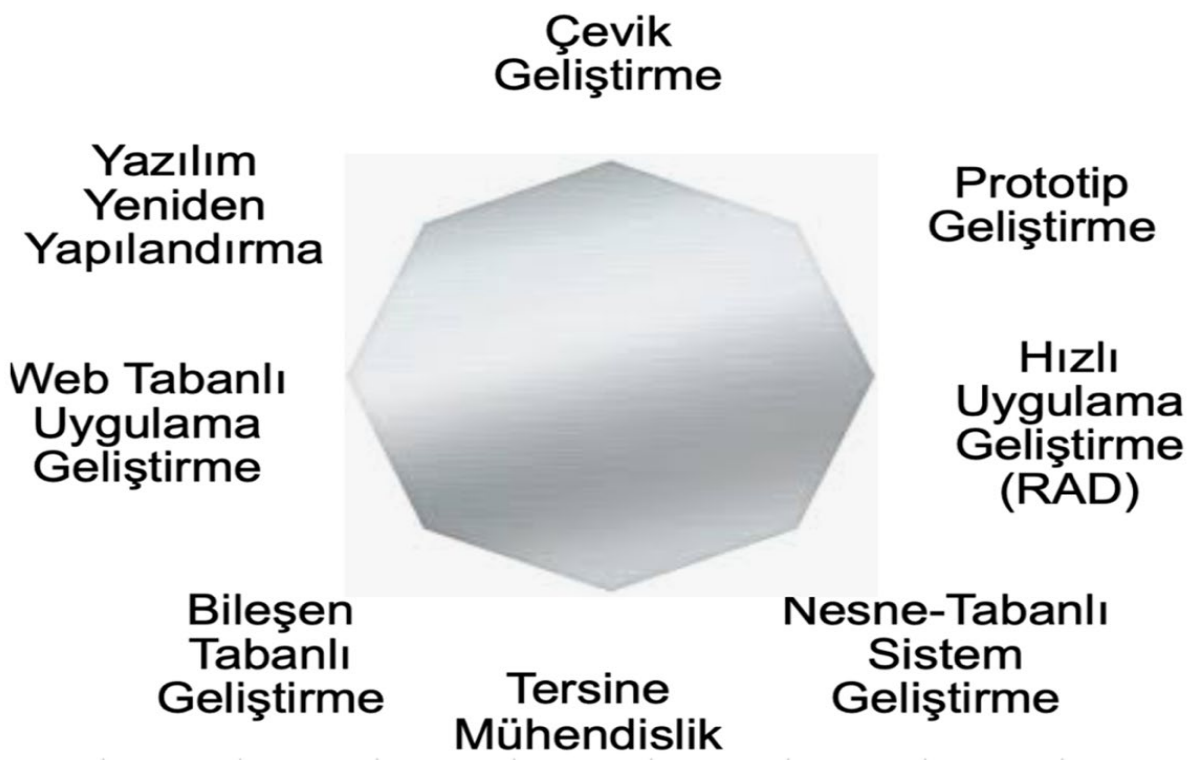
- Aşamalara göre anahtar teslimatlar ile bu teslimatlara proje personelinin atandığı doğrudan sorumluluklar,

- Anahtar teslimatların tamamlanması için vurgulanmış tarihleri olan bir proje takvimi,

- Her aşamanın tamamlanması için gereken kaynakları ve kaynak maliyetlerini tanımlayan ekonomik bir tahmin.

2.1.3. Yazılım Geliştirme Metodolojileri

Yazılım geliştirme metodolojileri aşağıdaki şemada verildiği üzere bünyesinde birçok unsuru barındırır.



Yazılım Geliştirme Metodolojisi Şeması

Çevik Geliştirme:

Çevik geliştirme, şelale metodolojisinin eksiklerini gidermek ve proje sürecine yeni bir yaklaşım kazandırmak için geliştirilmiştir. 2001 yılında Amerika'da yayınlanan Agile Manifestosu, bu metodolojinin prensiplerini ortaya koymuştur. Çevik geliştirme, karmaşık sistemler geliştirmenin geleneksel olmayan bir yolunu benimseyen benzer geliştirme süreçleri ailesidir. Bu süreçler, geliştirilmekte olan sistemdeki veya geliştirmeyi gerçekleştiren projedeki değişiklikleri esnek bir şekilde ele almak üzere tasarlandıkları için "*çevik*" olarak adlandırılır. Çevik yazılım geliştirme; yinelemeli (iterational) olarak yazılım geliştirme sürecinin aktivitelerinin uygulanması ve tedrici (incremental) olarak yazılım ürününün doğru bir şekilde, değişime açık, Çevik Birliği (Agile Alliance) oluşturduğu değerler ışığında prensiplerini uygulayarak geliştirilmesidir. Çevik geliştirme, projenin bir kere yapıp bitirmekten ziyade, projeyi en basit haliyle kullanıcıya sunmak ve kullanıcıdan gelecek dönüşler doğrultusunda projeyi sürekli olarak geliştirme amacı taşır.

Çevik yazılım geliştirme, her iterasyonda müşteriden alınan geri bildirimlerle yönlendirilir. Bu metodoloji, proje boyunca gereksinimlerdeki değişikliklere hızlı bir şekilde adapte olmayı sağlar. Bu esneklik, geliştirme süreci boyunca müşteri beklentilerini en iyi şekilde karşılamak amacıyla gereksinimlerin sürekli olarak gözden geçirilmesini ve yeniden şekillendirilmesini sağlar. Ayrıca, bu yaklaşım, proje süresince iş ihtiyaçlarındaki değişikliklere hızla adapte olmak için sık sık gereksinim gözden geçirme ve öncelik sırasının yeniden düzenlenmesi gerekliliğini ortaya koyar. Bu, proje sürecinde dinamik bir esneklik sağlar, ancak buna rağmen zaman yönetimi ve kaynak tahsisi açısından zorluklar doğurabilir.

Çevik geliştirme metodolojisini kullanmak birçok farklı alanda projelere pozitif değerler katar. Bunlara aşağıda yer verilmektedir:

Müşteri Odaklılık: Çevik geliştirme süreçleri, müşteri ihtiyaçlarına odaklanır. Müşteri geri bildirimleri sürekli olarak alınır ve bu geri bildirimlere dayalı olarak değişiklikler yapılır. Bu sayede müşterinin ihtiyaçlarına daha iyi yanıt veren bir ürün geliştirilir.

Ekip İşbirliği: Çevik geliştirme, tüm proje ekibinin sürekli işbirliği içinde çalışmasını teşvik eder. Yazılım geliştirme ekibi, yazılımın her aşamasında bir araya gelir, sorunları çözer ve birlikte kararlar alır.

Küçük İterasyonlar: Çevik yaklaşım, büyük ve karmaşık projeleri daha küçük, yönetilebilir parçalara böler. Her iterasyon, belirli bir süre içinde tamamlanabilir ve işlevsel bir parça ürün sunar. Bu, hızlı geri bildirim almayı ve esneklik sağlamayı kolaylaştırır.

Sürekli İyileştirme: Çevik geliştirme süreçleri, sürekli olarak kendini iyileştirme fırsatlarına odaklanır. Geri bildirimler ve deneyimler üzerinden sürekli olarak geliştirme yapılır.

Değişime Açıklık: Çevik yaklaşım, değişen gereksinimlere ve koşullara hızlı bir şekilde adapte olmayı teşvik eder. Bu nedenle, projeler daha az tahmin edilebilirlikle başa çıkabilir.

İletişim ve Dokümantasyon: Çevik geliştirme, iletişimi ve dokümantasyonu önemser, ancak gereksiz ayrıntılardan kaçınır. İletişim açık ve düzenli olmalıdır, ve dokümantasyon gerektiği kadar yapılmalıdır.

Test Odaklı Geliştirme: Çevik ekipler, kod yazmadan önce testleri düşünür ve testlerin yazılmasına büyük önem verir. Bu, ürünün kalitesini artırır ve hataların erken tespit edilmesini sağlar.

Sürdürülebilirlik: Çevik geliştirme, uzun vadeli sürdürülebilirliği hedefler. Sürekli olarak çalışabilir ve güncellenebilir bir ürünün geliştirilmesine odaklanır.

Prensiplere Dayalı Yaklaşım: Çevik geliştirme, Agile Manifesto adı verilen belirli prensiplere dayalı olarak uygulanır. Bu prensipler, özgürlük ve sorumluluğu vurgular.

Çevik Çerçeveler: Çevik geliştirme için birçok çerçeve (framework) ve yöntem bulunur, en popülerleri Scrum, Kanban ve Extreme Programming (XP) gibi. Bu çerçeveler, çevik prensipleri ve uygulamalarını belirli bir yapı içinde düzenler.

Çevik yazılım geliştirmenin avantajları:

- Kısa toplantılar ve yüz yüze iletişim felsefesi ile çevik yöntemlerde uygulanan insan ve iletişim merkezli süreçler sayesinde proje elemanlarının moral ve motivasyonu artar, değerli bilgi ve görüşlerin paylaşımı ile üretkenlik seviyeleri yükselir.

- Kısa süreli yinelemeler ve yineleme sonucunda teslim edilen çalışır ürün sayesinde projenin ilerlemesi daha sağlıklı bir şekilde gözlemlenir ve proje riski azalır, muhtemel hatalar daha erken tespit edilir, teslimat öngörülebilirliği artar ve ürün pazara daha çabuk ulaşır.

- Çevik yöntemlerin felsefesi gereği proje süresince değişikliklerin hoş karşılanması ve hatta teşvik edilmesi nedeniyle ve yinelemeli süreç yapısı sayesinde esneklik yüksektir ve değişen şartların yönetilmesi kolaydır.

- Müşteri veya kullanıcıların proje sürecinde proje elemanları ile yüz yüze iletişim kurabilmeleri sayesinde iş süreçleri ile yazılım geliştirme süreçlerinin entegrasyon seviyesi yükselir, istenen özelliklerin tam olarak ifade edilebilmesiyle yazılımın fonksiyonel kalitesi artar ve müşteri memnuniyeti sağlanır.

- Genel olarak üretilen yazılımın sadece ürün özellikleri dokümantasyona çevrilir ve diğer süreçler hariç tutulur. Hafif dokümantasyon iş yükünü azaltır.

- Çevik yöntemler, uygulanan safhalar arası geçişlere ve geri dönüşlere müsait olduğu için süreç devam ederken tespit edilen geriye dönük aksaklıklar kolaylıkla düzeltilebilir.

- Çevik yaklaşım, yazılım geliştirme sürecinin her aşamasında müşteri veya kullanıcı geri bildirimlerini dikkate almayı teşvik eder. Bu sayede, müşteri beklentilerine daha iyi cevap verilebilir ve yazılımın gereksinimlere daha iyi uyum sağlaması mümkün olur. Hızlı geri bildirim döngüleri, yazılımın daha iyi şekillendirilmesine yardımcı olur.

- Çevik yaklaşım, kaynakların daha verimli bir şekilde kullanılmasını sağlar. Proje ekibi, sürekli olarak öncelikli işleri belirleyerek ve bu işlere odaklanarak zaman ve kaynak israfını önler. Böylece,

sınırlı kaynaklar en iyi şekilde değerlendirilir ve projenin başarısı için gereken önemli işlere odaklanılır. Bu, proje maliyetlerini düşürebilir ve bütçe kontrolünü sağlayabilir.

Çevik yazılım geliştirmenin dezavantajları:

- Değişime açık olması nedeniyle projenin kaynak ve kapsam planlaması zordur.

- Çevik yöntemlerde süreçler iletişim merkezli olduğu için proje elemanlarının birbirleriyle veya müşteriyle aynı ortamda bulunmaması, proje gelişimini olumsuz etkiler. Video konferans gibi uzaktan iletişim kurabilecek teknolojiler kullanılabilir fakat bu da gerekli altyapı ve teknoloji nedeniyle maliyetleri arttırır.

- Alt yüklenicinin teklif verebilmesi ve bir kontrat yapılabilmesi için istenenlerin kesin olması ve değişmemesi gerekir. Çevik yöntemlerin belirsiz ve değişken ortamı alt yüklenici uygulamalarını zorlaştırır.

- Fonksiyonel olarak bütünlük gerektiren büyük ve karmaşık sistemler yinelemeler için uygun parçalara bölünemeyebilir.

- Zaman ayıramama veya isteksiz olma gibi nedenlerle müşteri veya son kullanıcıların proje içerisinde olma oranı düşükse, gereksinimler doğru biçimde tanımlanamayabilir.

- "*Gerektiği kadar*" anlayışı ile hazırlanan dokümantasyonun yetersiz olması durumunda projeye sonradan katılan geliştiriciler ve bakım safhaları olumsuz etkilenir.

- Takımlar kendi kendini organize ettiği için bireylerin yetenekli, tecrübeli ve çevik süreçlere hâkim olması gerekir. Yeteneksiz, tecrübesiz çalışanlar takım performansını düşürerek proje gelişim sürecini olumsuz etkiler.

- Önceliklerini tam olarak belirleyemeyen ve yapılacaklar listesindeki öncelikleri sık sık değiştirmek isteyen müşteriler planlamaları olumsuz etkileyebilir.

- Çevik yöntemlerle, spesifik problemler veya fonksiyonlara göre tanımlanan gereksinimleri karşılayacak yeterlikte çözüm geliştirildiği için, ürünlerin diğer projelerde yeniden kullanılabilirliği olmaz.

- Yazılım hatasının insan hayatını tehlikeye attığı veya büyük maddi zararlar doğurduğu hata toleransı olmayan sistemler için çevik yöntemlerin test ve kalite kontrol yöntemleri yeterli olmayabilir.

- Çevik yöntemlerde müşteri faydası ön planda tutulur fakat bu genel olarak müşterinin istediği fonksiyonların karşılanmasıdır. Yalın geliştirmenin bir kalıtımı olarak basitlik felsefesi uygulandığı için kullanıcı ara yüzü ve kullanıcı deneyimi gibi kullanıcı tatmini ile ilgili konular göz ardı edilebilir.

- Çevik geliştirme, projenin başlangıcında tüm gereksinimlerin belirsiz olduğu kabul eder, bu nedenle maliyet tahminleri daha zor olabilir. Projede sürekli değişiklikler yapılması maliyetleri kontrol etmeyi zorlaştırabilir.

- Çevik yaklaşım, projenin sürekli olarak izlenmesini ve değerlendirilmesini gerektirir. Bu, proje yöneticileri ve ekip üyeleri için daha fazla iş yükü anlamına gelebilir.

- Değişikliklere açık bir yaklaşım olan çevik geliştirme, beklenmedik risklerin ortaya çıkmasına neden olabilir. Bu risklerin zamanında tanımlanması ve ele alınması önemlidir.

- Ekip üyelerinin ve organizasyonun çevik yaklaşımı anlamaları ve uygulamaları zaman alabilir. Bu, başlangıçta verimliliği düşürebilir.

- Bazı projelerde müşteri kabul testlerinin tamamlanması ve müşterinin onay vermesi gerekebilir. Bu süreç, projenin tamamlanmasını geciktirebilir.

- Çevik geliştirme, hızlı iterasyonları teşvik eder, ancak yazılım güvenliği için gerekli testleri ve değerlendirmeleri ihmal etme riski vardır. Bu, güvenlik açıklarına yol açabilir.

- Proje süreçleri ve ilerleme, bazen geleneksel yöntemlere göre daha az şeffaf olabilir. Bu, üst düzey yöneticiler veya dış paydaşlar için projeyi izlemeyi zorlaştırabilir.

- Sürekli değişen gereksinimler, ekiplerde ve müşterilerde duygusal yorgunluğa neden olabilir. Proje ekibi, sürekli uyarlanma gerektiğinden bazen stres yaşayabilir.

Prototip Geliştirme (Evrimsel Geliştirme):

Prototip geliştirme, bir tasarımın çeşitli yönlerinin test edilmesi, fikirler veya özelliklerin gösterilmesi ve erken kullanıcı geri bildiriminin toplanması için çalışan bir modeli (prototip) hızla bir araya getirme sürecidir. Prototipleme, tavsiye ve eleştiri için kullanıcıya nihai sistemin bir modelini temsil etmek için programlanmış simülasyon tekniklerini kullanır. Vurgu, son kullanıcı ekranları ve raporları üzerindedir. Bu sadece bir model olduğu için iç kontroller öncelikli bir öğe değildir.

Prototipleme, bilgi sistemi kurma çalışmasına belirli bir soruna yönelik bir program ile başlamaktır. Temel olarak, prototip, sistem kullanıcılarına, sistemin tamamlanmış halinin ön görünümünü sağlamaktadır.

İşletme bazında düşünülecek olursa, belli bir bölüm seçilmekte ve bu bölüme özgü sorunu çözmek üzere bilgi sistemi oluşturulmaktadır. Bu sistem sadece o bölüme hizmet vermektedir.

İlerleyen süreçte ihtiyaç duyulduğunda diğer bölümlerde de benzeri programlar oluşturulmaya başlanıp, bu programın parçalarının bir araya getirilmesi veya ilk yazılan programın geliştirilmesi ile istenilen bilgi sistemine ulaşılabilmektedir.

Prototip geliştirmenin avantajları:

- Kullanıcıyı Alıştırma: Prototipleme yaklaşımının önde gelen amacı, kullanıcı ihtiyaçlarını büyük ölçüde karşılayan sistemler geliştirilmesidir.

- Hızlı Geliştirme Zamanı: Klasik yaklaşımda, anlamlı sonuçlara ulaşılması ve sistemi faaliyete geçirmek için yıllar gerekebilirken, bu yaklaşımda anlamlı sonuçlar elde edebilmek için kısa bir süre (birkaç hafta veya ay) yeterli olmaktadır.

- Daha Az Hata: Prototipleme yaklaşımı, hataların önceden belirlenmesini sağlamaktadır. Klasik yaklaşımla ise, hatalar ancak yıllar süren geliştirme sürecinin sonunda belirlenebilmektedir.

- Daha Çok Değişim Fırsatı: Prototipleme ile yöneticiler, geliştirilmekte olan her bir alt sistem veya komponentin çıktıları görebilmekte ve bu çıktılarla çalışabilmektedir. Bu sayede, yöneticiler geliştirme süreci içerisinde değişim önerilerinde bulunabilmektedir.

- Daha İyi İşbirliği: Prototip geliştirmenin sürekli geri bildirim döngüsü, tasarım ekibi, geliştiriciler ve paydaşlar arasında daha yakın işbirliği ve işbirliği sağlar. Bu, proje ekibinin birlikte çalışarak daha iyi sonuçlar elde etmesini kolaylaştırır.

- Maliyet Tasarrufu: Prototipler, tasarım hatalarının ve eksikliklerinin erken aşamalarda tespit edilmesine yardımcı olduğu için sonraki aşamalarda düzeltilmesi gereken sorunların maliyetini azaltabilir. Ayrıca, prototipler son ürünün tasarımının doğru olduğundan emin olunmadan büyük miktarda kaynağın harcanmasını engeller.

- Daha İyi Proje Yönetimi: Prototip geliştirmenin sürekli izleme ve değerlendirme gerektiren doğası, proje yöneticilerine daha fazla kontrol ve daha iyi proje yönetimi sağlar. Projenin ilerlemesini daha iyi takip etmelerine ve gerektiğinde düzeltici önlemler almalarına yardımcı olur.

- Müşteri Memnuniyeti: Prototipler, kullanıcıların veya müşterilerin ürün veya sistem tasarımına daha fazla katılımını teşvik eder. Bu, son ürünün müşteri ihtiyaçlarına daha iyi uygun olmasını sağlar ve sonuçta müşteri memnuniyetini artırır.

- Daha Kolay Eğitim: Prototipler, son ürünün nasıl kullanılacağına dair eğitim materyali olarak kullanılabilir. Kullanıcılar ve çalışanlar, prototipleri kullanarak sistemi daha iyi anlayabilirler.

- Rekabet Üstünlüğü: Prototip geliştirme, daha hızlı bir şekilde yeni ürünler veya özellikler sunmayı mümkün kılar. Bu, şirketin rekabetçi bir avantaj elde etmesine yardımcı olabilir.

- Daha İyi Risk Yönetimi: Prototipler, projenin risklerini daha erken aşamalarda tanımlamanıza ve yönetmenize olanak tanır. Bu, projenin daha az beklenen sorunlarla karşılaşmasına yardımcı olur.

Prototip geliştirmenin dezavantajları:

- Sistem Geliştiriciler ile Yönetim Arasındaki Ortaklık İhtiyacı: Yönetimin aktif olarak projeye katılması, bazı yöneticiler tarafından ayrılmak istenmeyen bir zamana ihtiyaç duyulmasına neden olmaktadır.

- Toplam Geliştirme Maliyetinin Yüksek Olabilmesi: Yöneticilerin sürekli olarak sistemde değişiklik ihtiyacı mevcutsa bu bir dezavantaj olarak ortaya çıkmaktadır.

- Yeni veya Geliştirilmiş Sistemin Bilgisayar Kaynaklarını Etkin Olarak Kullanamaması: Prototipleme yaklaşımında, kullanıcı ihtiyaçlarına çok fazla önem verilirken, çeşitli bileşenlerin kullanımına daha az önem verilmektedir. Sistemlerin daha hızlı ve ucuz hale gelmesiyle, bu donanım limitleri daha önemli bir hal almaktadır.

- Sürekli Değişim ve Belirsizlik: Prototip geliştirme, sürekli değişen gereksinimlere ve tasarım değişikliklerine açık bir yaklaşımı teşvik eder. Bu, proje ekibinin ve yönetimin sürekli olarak yeni değişiklikleri yönetmesini ve kaynakları bu değişikliklere göre yeniden tahsis etmesini gerektirebilir.

- Kapsam Kontrolü Zorluğu: Sürekli değişen gereksinimler nedeniyle proje kapsamının kontrol altında tutulması zorlaşabilir. Bu, proje süresi ve maliyetinin tahmin edilmesini güçleştirir.

- Yazılım Mimarisi Zorlukları: Prototip geliştirme, son ürünün yazılım mimarisini oluşturmak için sık kullanılan yöntemlerden farklıdır. Bu nedenle, prototipin son ürünle uyumlu hale getirilmesi gerekebilir, bu da ek zaman ve kaynak gerektirebilir.

- Müşteri Beklentileri: Prototipler genellikle son üründen daha az işlevseldir ve müşterilere gerçek sistemle ilgili yanlış beklentiler oluşturabilir. Bu, müşteri memnuniyetsizliğine veya anlaşmazlıklara neden olabilir.

- Veri Güvenliği: Prototip geliştirme sırasında, hassas verilerin kullanılması veya korunması önemli bir zorluk olabilir. Veri güvenliğine yönelik yeterli önlemler alınmazsa, bu verilerin sızması riski vardır.

- Dokümantasyon Eksikliği: Prototip geliştirmenin odak noktası, çalışan bir prototip oluşturmak olduğu için genellikle geliştirme süreciyle ilgili eksik veya yetersiz dokümantasyon olabilir. Bu, sonraki aşamalarda bakım, eğitim ve gelecekteki değişiklikler için zorluklar yaratabilir.

- Eğitim İhtiyacı: Prototip geliştirmeyi kullanmak, proje ekibi ve paydaşlar arasında bu yaklaşımın nasıl çalıştığını anlamayı gerektirir. Bu nedenle ek eğitim ve bilinçlendirme ihtiyacı doğar.

Hızlı Uygulama Geliştirme (Rapid Application Development-RAD):

Hızlı uygulama geliştirme, 1970'lerde tasarlanmış; ancak resmi olarak 1991'de James Martin tarafından tanıtılmıştır. Hızlı uygulama geliştirme, iyi tanımlanmış bir metodoloji dahilinde, bir dizi kanıtlanmış uygulama geliştirme tekniğini kullanarak geliştirme maliyetlerini düşürürken ve kaliteyi korurken, işletmelerin stratejik olarak önemli sistemleri daha hızlı geliştirmelerini sağlayan bir metodolojidir. Hızlı uygulama geliştirme, sürekli müşteri geri bildirimleriyle sık yinelemeler ve onaylar kullanılarak hızlı bir şekilde uygulama geliştirmeye odaklanır. Çevik ve hızlı prototip sürümlerine öncelik vermesi sayesinde RAD, özel uygulamalar gibi öğelerin oluşturulması sürecinde uzun süreli planlama ve başlangıçta belirtilen tek bir gereksinim grubu yerine yazılım kullanılabilirliğini, kullanıcı geri bildirimlerini ve hızlı teslimi öne çıkarır.

James Martin'in RAD'ye yaklaşımı süreci dört farklı aşamaya böler:

1- Gereksinim Planlama Aşaması: SDLC'nin sistem planlama ve sistem analiz aşamalarının unsurlarını birleştirir. Kullanıcılar, yöneticiler ve BT personeli, iş ihtiyaçları, proje kapsamı, kısıtlamalar ve sistem gereksinimleri üzerinde tartışır ve anlaşırlar. Ekip, kilit konular üzerinde anlaştığında ve devam etmek için yönetim yetkisini aldığında sona erer.

2- Kullanıcı Tasarımı Aşaması: Bu aşamada kullanıcılar sistem analistleriyle etkileşime girer ve tüm sistem süreçlerini, girdilerini ve çıktılarını temsil eden modeller ve prototipler geliştirir. RAD grupları veya alt grupları, kullanıcı ihtiyaçlarını çalışan modellere dönüştürmek için tipik olarak ortak uygulama geliştirme (JAD) teknikleri ve bilgisayar destekli yazılım mühendisliği (Computer Assisted

Software Engineering, CASE) araçlarının bir kombinasyonunu kullanır. Kullanıcı tasarımı, kullanıcıların ihtiyaçlarını karşılayan sistemin çalışan bir modelini anlamalarını, değiştirmelerini ve nihayetinde onaylamalarını sağlayan sürekli etkileşimli bir süreçtir.

3- İnşaat Aşaması: SDLC'ye benzer program ve uygulama geliştirme görevine odaklanır. Ancak RAD'de kullanıcılar katılmaya devam eder ve gerçek ekranlar veya raporlar geliştirildikçe değişiklik veya iyileştirmeler önerebilir. Görevleri programlama ve uygulama geliştirme, kodlama, birim entegrasyonu ve sistem testidir.

4- Geçiş Aşaması: Veri dönüştürme, test etme, yeni sisteme geçiş ve kullanıcı eğitimi dahil olmak üzere SDLC uygulama aşamasındaki son görevlere benzer. Geleneksel yöntemlerle karşılaştırıldığında, tüm süreç sıkıştırılır. Sonuç olarak, yeni sistem çok daha kısa sürede kurulur, teslim edilir ve işletime alınır.

RAD'nin avantajları aşağıdakileri içerir:

- Daha İyi Kalite: Kullanıcıların gelişen prototiplerle etkileşime girmesini sağlayarak, bir RAD projesinden elde edilen iş işlevselliği, genellikle bir şelale modeliyle elde edilenden çok daha yüksek olabilir. Yazılım daha kullanışlı olabilir ve geliştiricileri ilgilendiren teknik sorunlardan ziyade son kullanıcılar için kritik olan iş sorunlarına odaklanma şansı daha yüksektir. Ancak bu, güvenlik ve taşınabilirlik dahil, genellikle işlevsel olmayan gereksinimler (AKA kısıtlamaları veya kalite özellikleri) olarak bilinen diğer kategorileri hariç tutar.

- Risk Kontrolü: RAD ile ilgili literatürün çoğu hız ve kullanıcı katılımına odaklansa da RAD'nin doğru bir şekilde yapılmasının diğer kritik bir özelliği de risk azaltmadır. Bir RAD yaklaşımı, temel risk faktörlerine erkenden odaklanabilir ve sürecin ilk bölümünde toplanan ampirik kanıtlara dayanarak bunlara uyum sağlayabilir. Örneğin, sistemin en karmaşık parçalarından bazıları prototiplemenin karmaşıklığıdır.

- Zamanında ve Bütçe Dahilinde Tamamlanan Daha Fazla Proje: Artımlı birimlerin geliştirilmesine odaklanarak, büyük şelale projelerini sürdüren yıkıcı başarısızlık şansı azalır. Şelale modelinde, tüm sistemin radikal bir şekilde yeniden düşünülmesini gerektiren 6 ay veya daha fazla analiz ve geliştirmeden sonra bir sonuca varmak yaygındı. RAD ile bu tür bilgiler keşfedilebilir ve süreç içinde daha erken harekete geçilebilir.

- Daha Yüksek Müşteri Memnuniyeti: RAD, müşteri katılımını ve geri bildirimini teşvik eder. Kullanıcılar, prototipleri inceleyerek ve etkileşimde bulunarak daha erken aşamalarda yazılımın gelişimini izleyebilirler. Bu, müşterilerin ihtiyaçlarına daha iyi cevap veren bir ürünün daha hızlı bir şekilde teslim edilmesine yardımcı olur ve müşteri memnuniyetini artırır.

- Hızlı Değişiklikler ve İterasyonlar: RAD, değişen gereksinimlere hızlı bir şekilde adapte olabilen bir yaklaşım sunar. Prototipler üzerinde yapılan değişiklikler, hızlıca uygulanabilir ve test edilebilir. Bu, projenin esnekliğini artırır ve değişikliklere uyum sağlama yeteneğini geliştirir.

- Daha İyi İşbirliği: RAD, proje ekibi üyeleri arasında daha yakın işbirliğini teşvik eder. Geliştiriciler, analistler ve kullanıcılar, sürekli olarak iletişim kurarlar ve birlikte çalışırlar. Bu, projenin daha iyi anlaşılmasını ve işbirliğini artırır.

- Daha Kısa Proje Süreleri: RAD, geleneksel şelale modeline göre daha kısa projelerin tamamlanmasına olanak tanır. İteratif ve artımlı yaklaşım, yazılımın hızlı bir şekilde teslim edilmesini sağlar.

- Maliyet Tasarrufu: RAD, projenin daha hızlı bir şekilde tamamlanmasını ve kaynakların daha etkili bir şekilde kullanılmasını teşvik eder. Bu da genellikle projenin maliyetlerini düşürebilir.

RAD'nin dezavantajları aşağıdakileri içerir:

- Çoğu taraf için RAD, deneyimli profesyonellerin çalışma biçimlerini yeniden düşünmelerini gerektiren yeni bir yaklaşımdır. İnsanlar hemen hemen her zaman değişime karşıdır ve yeni araçlar veya yöntemlerle üstlenilen herhangi bir projenin, yalnızca ekibin öğrenme gereksinimi nedeniyle ilk seferde başarısız olma olasılığı daha yüksektir.

- Normal çalışmada genellikle son kullanıcı tarafından görülmeyen, işlevsel olmayan gereksinimlere vurgu yapılabilir.

- Neredeyse tüm RAD yaklaşımlarının ortak noktası, kullanıcılar ve geliştiriciler arasında tüm yaşam döngüsü boyunca çok daha fazla etkileşim olmasıdır. Bu da fazla zaman kaybına neden olacaktır.

- Daha az kontrol mevcuttur. RAD'nin avantajlarından biri, esnek ve uyarlanabilir bir süreç sağlamasıdır. İdeal olan, hem sorunlara hem de fırsatlara hızla uyum sağlayabilmektir. Esneklik ve kontrol arasında kaçınılmaz bir değiş tokuş vardır. Birinin fazla olması diğerinin daha az olması anlamına gelir. Bir projede (örneğin hayati önem taşıyan bir yazılım) değerlerin çeviklikten daha fazla kontrolü varsa RAD uygun değildir.

- Tasarımı nispeten daha kötüdür. Prototiplere odaklanma, bazı durumlarda geliştiricilerin sürekli olarak bireysel bileşenlerde küçük değişiklikler yaptığı ve daha iyi bir genel tasarımla sonuçlanabilecek sistem mimarisi sorunlarını göz ardı ettiği "*hack ve test*" metodolojisiyle sonuçlanan çok ileri götürülebilir. Bu, özellikle James Martin'inki gibi sistemin kullanıcı arayüzüne çok fazla odaklanan metodolojiler için bir sorun olabilir.

- Ölçeklenebilirlik eksikliği mevcuttur. RAD tipik olarak küçük ve orta ölçekli proje ekiplerine odaklanır. Yukarıda belirtilen diğer sorunlar (daha az tasarım ve kontrol), çok büyük ölçekli sistemler için bir RAD yaklaşımı kullanırken özel zorluklar ortaya çıkarır.

- RAD, hızlı prototip oluşturmayı teşvik ettiği için, veri güvenliği konusu önemli bir sorun olabilir. Prototipler hassas verileri içerebilir ve bu verilerin korunması zor olabilir.

- RAD, hızlı gelişim ve prototipleme üzerine odaklandığı için uzun vadeli bakım ve sürdürülebilirlik konularını ikinci plana atabilir. Bu, ilerleyen aşamalarda yazılımın bakımının daha zor ve maliyetli hale gelmesine neden olabilir.

- RAD, değişken gereksinimlere ve prototiplerin sürekli olarak değişmesine dayalıdır. Bu, bazı durumlarda projenin ne zaman tamamlanacağını veya ne kadar maliyetli olacağını tahmin etmeyi zorlaştırabilir.

- RAD'ı etkili bir şekilde uygulamak için deneyimli ve yetenekli bir ekip gereklidir. Bu, yeni başlayanlar veya daha az deneyime sahip ekipler için daha zorlu olabilir.

- RAD, sürekli iletişimi ve işbirliğini teşvik eder, ancak bazen bu, farklı coğrafi bölgelerdeki ekipler veya farklı dilleri konuşan ekipler arasında iletişim zorluklarına neden olabilir.

- Prototipler, son ürünün bir ön izlemesi olarak kullanıldığından, kullanıcılar bazen prototipin son üründen farklı olduğunu unutabilirler. Bu, yanıltıcı beklentilere yol açabilir.

Nesneye Yönelik Sistem Geliştirme (Object Oriented System Development-OOSD):

Nesneye yönelik sistem geliştirme (OOSD) analistlerin, geliştiricilerin ve programcıların bir sistemin daha büyük mantıksal parçalarını dikkate almalarına ve programlama sürecini netleştirmelerine izin veren, verileri ve prosedürleri nesneler halinde gruplandıran bir programlama tekniğidir. Klasik yaklaşımın aksine, veri merkezli, sınıf modelleri üzerinde geliştirilen bir yaklaşımdır. Bu yöntem kullanıldığında kodların ve modüllerin yeniden kullanılabilirliği kolaylaşmakta, geliştirme süresi kısalmakta, üretkenlik artmakta, yazılım kalitesi yükselmekte, anlaşılabilirdik kolaylaşmaktadır.

OOSD, sınırsız çeşitlilikteki verilerin yönetimine, karmaşık ilişkileri modelleme becerisine ve değişen bir ortamın taleplerini karşılama becerisine izin verir.

Bu yaklaşımın bazı avantajları ve özelliklerine aşağıda yer verilmiştir:

- Modülerlik ve Yeniden Kullanılabilirlik: OOSD, kodları nesneler halinde gruplandırarak modüler bir yapı oluşturur. Bu, yazılımın farklı projelerde veya farklı bileşenlerde yeniden kullanılmasını kolaylaştırır. Modüllerin bağımsız olması, bakımı ve güncellemeyi de daha kolay hale getirir.

- Üretkenlik Artışı: Nesnel programlama, daha hızlı ve etkili bir şekilde kod yazmayı teşvik eder. Birçok dilde hazır kütüphaneler ve çerçeveler bulunur, bu da geliştirme sürecini hızlandırabilir.

- Yazılım Kalitesi: OOSD, daha iyi kod organizasyonu ve anlaşılabilirliği teşvik eder. Bu da daha az hata, daha kolay hata ayıklama ve daha iyi bir yazılım kalitesi anlamına gelir.

- Değişikliklere Duyarlılık: OOSD, değişen gereksinimlere uyum sağlama yeteneği ile tanınır. Bir nesne tabanlı sistemde, yeni gereksinimler eklemek veya mevcutları değiştirmek genellikle daha sorunsuz bir şekilde yapılabilir.

- Veri Yönetimi ve Karmaşıklık İlişkileri: OOSD, karmaşık veri yapılarını ve ilişkileri modellemek için uygun bir araç sağlar. Bu, büyük ve karmaşık verilerle çalışmanın gerektiği projeler için önemlidir.

- İşbirliği Kolaylığı: OOSD, bir ekip içindeki farklı üyeler arasında işbirliğini kolaylaştırır. Sınıfların ve nesnelerin tanımlanması, herkesin aynı terminolojiyi kullanmasını sağlar.

- Büyüklük ve Karmaşıklıkla Başa Çıkma: OOSD, büyük ve karmaşık sistemlerin geliştirilmesini daha yönetilebilir hale getirebilir. Bu, büyük projelerdeki karmaşıklığı azaltabilir.

- OOSD, günümüz yazılım geliştirme alanında yaygın olarak kullanılan bir yaklaşım olmuştur ve birçok popüler programlama dilinin temelini oluşturmuştur. Bu nedenle, yazılım geliştirme projelerinde OOSD prensiplerini kullanmak, genellikle daha etkili ve sürdürülebilir sonuçlar elde etmeye yardımcı olabilir.

OSD yaklaşımının dezavantajları aşağıdakiler olabilir:

- Öğrenme Eğrisi: OOSD, geleneksel programlama paradigmasından farklı bir yaklaşım gerektirir. Bu nedenle, daha önce bu yöntemi kullanmamış geliştiriciler için öğrenme eğrisi dik olabilir.

- Tasarım Karmaşıklığı: OOSD, iyi bir nesne tabanlı tasarımın önemini vurgular. Ancak karmaşık ve yanlış tasarlanmış bir nesne tabanlı sistem, daha kötü bir sonuç verebilir.

- Bağımlılık Sorunları: Nesne tabanlı sistemlerde sınıflar ve nesneler arasındaki bağımlılıklar yönetilmelidir. Yanlış bir tasarım bu bağımlılıkları karmaşık hale getirebilir ve bakımı zorlaştırabilir.

- Performans Meseleleri: OOSD bazen ekstra katmanlar veya soyutlamalar ekleyebilir, bu da bazı durumlarda performans kaybına neden olabilir. Bu, özellikle kaynak sınırlamaları olan uygulamalar için önemli olabilir.

- Yeniden Eğitim Maliyeti: Bir organizasyonun OOSD'ye geçişi, personelinin bu yeni yaklaşımı öğrenmesini gerektirir. Bu da eğitim maliyetleri ve iş sürekliliği üzerinde etkiler yaratabilir.

- Fazla Soyutlama: OOSD, gereksinimlere göre gereksiz soyutlamaların eklenmesine yol açabilir. Bu, kodun anlaşılmasını zorlaştırabilir ve gereksiz karmaşıklığa neden olabilir.

- Uyum Sorunları: Eğer bir organizasyon daha önce geleneksel yöntemlerle çalışıyorsa, OOSD'ye geçiş süreci uyum sorunlarına neden olabilir. İnsanlar alıştıkları iş akışlarını ve yöntemleri yeniden düşünmek zorunda kalabilirler.

Bu dezavantajlar, OOSD'nin uygulanması sırasında dikkate alınması gereken faktörlerdir. Herhangi bir geliştirme yaklaşımının avantajları ve dezavantajları vardır ve proje gereksinimlerine, ekibin yeteneklerine ve organizasyonun hedeflerine bağlı olarak değerlendirilmelidir.

Bileşen Tabanlı Yazılım Geliştirme (Component-based Software Development-CBSD):

Bileşen tabanlı geliştirme, hizmetlerini tanımlanmış ara birimler aracılığıyla kullanıma sunan, iş birliği yapan yürütülebilir yazılım paketlerinden uygulamaları bir araya getirir. Geliştirme süresini ve maliyetini azaltır, kaliteyi artırır, modülerliği destekler ve yeniden kullanımı kolaylaştırır.

Yazılım mühendisliği ve yazılım endüstrisi büyük sistemleri oluştururken daha önceden hazırlanmış bileşenler kullanarak üretkenlik ve kaliteyi artırma fikrini benimsemiştir. Bu fikir bilgisayar bilimlerinde çoğu alanda kullanılan böl ve birleştir yöntemi ile uyuşmaktadır. Bu kapsamda ortaya atılan bileşen tabanlı yazılım geliştirme (CBSD), 1990'ların başında ortaya çıkmış yeniden kullanımı artıran yöntemlerden biridir. Bileşenler, çalışma anında mevcut bir yazılım sisteminin çalışmasını etkilemeden yeni özelliklerin eklenmesini sağlayan bağımsız yazılım birimleridir. CBSD, bu bileşenleri kullanarak ekle ve çalıştır mantığıyla çalışır. Bu yapı yazılım sistemlerinin bakımı sürecinde büyük faydalar

sağlamaktadır. Yeni gereksinimleri karşılamak için yeni bileşen eklemek veya var olan bileşenleri değiştirmek en uygun çözüm olarak görünmektedir. CBSD ile ilgili birkaç genel kural aşağıdaki gibi sıralanabilir:

- Ara yüz tabanlıdır. Bileşenler ara yüzü ve uygulamayı birbirinde ayırır ve uygulamanın detayını gizleyerek soyutlamayı sağlar.

- Mimari tabanlıdır. Daha önceden CBSD'ye göre tasarlanmış bir mimaride çalışabilirler, böylelikle yeni bileşen kolaylıkla diğer bileşenlerle birlikte çalışabilmektedir. Kullanılan mimariler genellikle MFC (Microsoft Foundation Class), CORBA, EJB gibi yapıların üzerinde kurulmaktadır. Bileşenler, ürüne özgü, alana özgü veya alan bağımsız olabilmektedir (CORBA Domain Objects gibi).

- Bileşenler için sınıflandırma şu şekilde yapılmıştır: Grafiksel kullanıcı arayüzü (GUI) bileşenleri en çok bulunan bileşenlerdendir. Bu bileşenlerde kod karmaşıklığı az olduğu için geliştirme maliyeti düşüktür. Örnek olarak; butonlar, metin alanları, veri gösterme için özelleştirilmiş tablolar gibi kullanıcı ara yüzleri verilebilir. Servis bileşenleri; daha karmaşık ve maliyeti yüksek bileşenlerdir. Uygulamaların ihtiyaç duyduğu ortak servisleri, veri tabanı erişimini sağlarlar. Örnek olarak; veri tabanı işlem servisleri verilebilir. Alana özgü bileşenler; geliştirmesi ve tekrar kullanımı en zor olan bileşenlerdir. Bileşenin kullanılacağı alan iyi tanımlanmış olmalıdır. Üretkenlik artışı çok fazladır. Örnek olarak; bir sigorta şirketi uygulaması için fatura, poliçe gibi bileşenlerin kullanılması verilebilir.

Geleneksel yazılım geliştirme yöntemlerinden farklı olarak CBSD'de yazılım geliştirme süreci bileşen geliştirme ve bileşen entegrasyonu olarak ikiye ayrılır. Bu ayrıma göre bileşenlerin uygulanabilirlik, genel bir bileşen olma, diğer bileşenlerle birlikte çalışabilirlik gibi özellikleri incelenir. Bir bileşen bir sisteme eklenmeden önce bazı uyarlamaların yapılması gerekebilir. Buna bileşenin optimizasyonu adı verilmektedir. Bu işlem yapılırken bileşene yapılan değişiklik sonucu bileşenin temel özellikleri etkilenmemelidir. Örnek olarak; bir menü 10 bileşenin mobil cihazlarda daha küçük boyutta ve çözünürlükte gösterilmesi menü bileşeninde bir düzenleme yapılarak sağlanabilir. Birbiriyle etkileşim içinde olan iki bileşen birleşerek yeni bir bileşen oluşturabilir. Buna bir bakıma uygulama çatısı da denilebilir. Heineman'a göre, bileşenlerin birbiriyle etkileşimi istemci/sunucu ve yayıncı/abone mimarileri ile sağlanabilmektedir. İlkinde bileşenler bir istemci gibi davranarak istediği bilgiyi dönen yöntem çağrımı yaparlar. İkincisinde ise bir bileşene kayıt olup ondan sürekli bildirimler alırlar. Bir bileşen modeli bu sayılan özellikleri gerçekleştirmiş olmalıdır. Bahsedilen seçme, kullanma ve entegrasyon zorluklarına rağmen bu yaklaşım sıkça kullanılmaktadır.

Aşağıda CBSD'nin uygulama alanlarına ve genel yapısına değinilmiştir:

- Uygulama Alanları: CBSD, genellikle büyük ve karmaşık yazılım projelerinde kullanılır. Özellikle işletim sistemleri, veritabanları, büyük ölçekli web uygulamaları ve finansal yazılımlar gibi sektörlerde yoğun olarak kullanılır. Bu alanlarda, CBSD'nin sağladığı modülerlik ve yeniden kullanılabilirlik büyük avantajlar sunar.

- Bağımsızlık ve Entegrasyon: CBSD, her bir bileşenin bağımsız olarak geliştirilmesini ve test edilmesini sağlar. Bu, geliştiricilerin belirli bir bileşeni daha önce geliştirdiğinden emin olmalarına ve ardından diğer bileşenlerle sorunsuz bir şekilde entegre etmelerine yardımcı olur.

- Standartlar ve Protokoller: CBSD projeleri genellikle belirli standartlara ve iletişim protokollerine dayalı olarak geliştirilir. Bu standartlar, bileşenlerin birlikte çalışabilirliğini ve uyumluluğunu sağlamak için kritik öneme sahiptir. Örneğin, Java EE veya .NET Framework gibi teknolojiler bu tür standartları uygulamada kullanılır.

- Yeniden Kullanılabilirlik: CBSD, yazılım bileşenlerinin yeniden kullanılabilirliğini teşvik eder. Bir bileşen bir projede kullanıldıktan sonra, aynı bileşen başka projelerde veya farklı uygulama alanlarında tekrar kullanılabilir. Bu, yazılım geliştirme sürecini hızlandırır ve maliyeti düşürür.

- Zorluklar: CBSD'nin uygulanması bazı zorluklarla karşılaşabilir. Özellikle, uygun bileşenlerin bulunması ve bunların projeye entegre edilmesi gereklidir. Ayrıca, bazı durumlarda bileşenlerin uyumlu bir şekilde çalışması için özel yazılım geliştirme çabaları gerekebilir.

- Veri Güvenliği: Özellikle hassas veri işleme gereksinimleri olan projelerde, CBSD'nin veri güvenliği konusunda özel önlemler gerektirebilir. Bu, veri bütünlüğünü ve gizliliğini korumak için özel önlemler gerektirir.

Web Tabanlı Uygulama Geliştirme:

Sunucu ve istemci arasında işlem gören, istemci tarafından dosya/uygulama yüklenmesi yapmadan işlev gösteren tarayıcı tabanlı çalışan uygulamalar web tabanlı uygulama olarak nitelendirilir.

Bu yaklaşım, işletmeler içinde ve arasında kod modüllerinin daha etkin entegrasyonunu sağlamak için XML dillerini (SOAP, WSDL, UDDI) kullanır. Web tabanlı uygulama geliştirmenin temel avantajları ve özelliklerine aşağıda yer verilmektedir:

- Platform Bağımsızlık: Web tabanlı uygulamalar, çoğu modern web tarayıcısı üzerinde çalışabilirler. Bu, kullanıcıların farklı işletim sistemlerinde ve cihazlarda aynı uygulamaya erişebilmelerini sağlar. Bu, mobil cihazlar, masaüstü bilgisayarlar ve tabletler dahil olmak üzere farklı platformlarda tutarlı bir deneyim sunma yeteneği anlamına gelir.

- Veri Güncelleme Kolaylığı: Web tabanlı uygulamalar, sunucu taraflı işlemler gerçekleştirerek kullanıcılara canlı ve güncel veri sunma yeteneği sağlar. Bu, kullanıcıların herhangi bir zamanda en son verilere erişmelerini ve güncel bilgilere dayalı kararlar almalarını kolaylaştırır.

- Kolay Dağıtım ve Güncelleme: Web tabanlı uygulamalar, kullanıcılara herhangi bir kurulum veya güncelleme gerektirmeden tarayıcıları üzerinden erişim sağlar. Bu, yazılım güncellemelerini merkezi bir şekilde yönetmeyi ve hızlı bir şekilde yeni özellikler eklemeyi kolaylaştırır.

- İşbirliği Kolaylığı: Web tabanlı uygulamalar, kullanıcıların aynı uygulamada işbirliği yapmalarını sağlar. Çevrimiçi doküman paylaşımı, çoklu kullanıcı erişimi ve sanal toplantılar gibi işbirliği araçlarına entegre edilebilirler.

- Güvenlik Odaklı Yaklaşım: Web tabanlı uygulamalar, genellikle güvenlik açısından dikkatlice tasarlanır ve geliştirilir. Veri şifreleme, kimlik doğrulama, oturum yönetimi ve yetkilendirme gibi güvenlik önlemleri uygulamak için kullanılırlar.

- Uzaktan Erişim: Web tabanlı uygulamalar, herhangi bir yerden ve herhangi bir cihazdan erişilebilir. Bu, uzaktan çalışma veya seyahat sırasında iş yapma esnekliği sağlar.

- Özelleştirme Yeteneği: Web tabanlı uygulamalar, kullanıcıların ihtiyaçlarına ve tercihlerine göre özelleştirilebilir. Kullanıcılar genellikle arayüzü kişiselleştirebilir veya farklı eklentileri entegre edebilirler.

- Veri Entegrasyonu: Web tabanlı uygulamalar, farklı veri kaynaklarından veri toplamayı ve bu verileri bir araya getirmeyi kolaylaştırır. Bu, işletmelerin daha fazla veri analitiği yapmasına ve daha bilinçli kararlar almasına yardımcı olur.

- Mobil Uyumluluk: Web tabanlı uygulamalar, mobil cihazlarda da iyi çalışacak şekilde tasarlanabilir. Bu, kullanıcıların mobil cihazlarını kullanarak uygulamalara erişmelerini ve kullanmalarını sağlar.

Yazılım Yeniden Yapılandırma:

Yazılım yeniden yapılandırma, mevcut sistemlerden bileşenlerin çıkarılmasını ve bu bileşenlerin yeni sistemler geliştirmek veya mevcut sistemlerin verimliliğini artırmak için yeniden yapılandırılmasını içeren bir süreçtir. Bu süreçte, mevcut yazılımların işlevsel (functional) ve/veya işlevsel olmayan (non-functional) niteliklerinin geliştirilmesi ile eski yazılım sistemlerinin dönüştürülmesinde, yeni işlevler kazandırılmasında kullanılan yöntem ve araçlar üzerine odaklanmaktadır.

Herhangi bir organizasyonda yapı, sistem, süreç ve uygulanan politikalarda hızlı ve radikal bir şekilde yeniden tasarım ve değişiklikler yapılarak organizasyonun daha yüksek performansa ulaşmasını amaçlayan yönetim tekniğidir. Böylece mevcut yazılım sistemleri, işlevselliklerini uzatmak için modernize edilebilir.

Yazılım yeniden yapılandırmanın avantajları ve önemli yönlerine aşağıda yer verilmektedir:

- Verimliliğin Artırılması: Mevcut yazılım sistemlerinin bileşenlerinin yeniden yapılandırılması, daha verimli ve hızlı çalışan sistemlerin oluşturulmasına olanak tanır. Bu, organizasyonun iş süreçlerini daha etkili bir şekilde yönetmesine ve hızlı tepki vermesine yardımcı olabilir.

- Maliyet Tasarrufu: Yazılım yeniden yapılandırma, genellikle yeni bir yazılım geliştirmekten daha ekonomik olabilir. Mevcut bileşenlerin kullanılması ve sadece ihtiyaç duyulan değişikliklerin yapılması maliyetleri azaltabilir.

- Modernizasyon: Mevcut yazılım sistemleri, eski veya eskimiş olabilir. Yeniden yapılandırma, bu sistemleri modern teknoloji ve standartlara uygun hale getirme fırsatı sunar. Bu da uzun vadeli sürdürülebilirlik ve rekabet avantajı sağlayabilir.

- Hızlı Uygulama Geliştirme: Yeniden yapılandırma, yeni yazılım geliştirmeye kıyasla daha hızlı bir yol olabilir. Mevcut bileşenlerin kullanılması, geliştirme sürecini hızlandırabilir ve yeni özelliklerin daha hızlı bir şekilde piyasaya sürülmesine olanak tanır.

- Daha İyi Uyum: Organizasyonun ihtiyaçları değiştikçe yazılım sistemlerini uyumlu hale getirmek önemlidir. Yeniden yapılandırma, yazılım sistemlerini daha iyi bir şekilde organizasyonun değişen gereksinimlerine uyarlayabilir.

- Risk Azaltma: Mevcut bir yazılım sistemi üzerinde yapılan değişiklikler, genellikle yeni bir yazılım geliştirme projelerine kıyasla daha az risk taşır. Çünkü mevcut sistemlerin işleyişi ve performansı genellikle bilinir.

- Varlık Yönetimi: Yazılım yeniden yapılandırma, organizasyonun yazılım varlıklarını daha etkili bir şekilde yönetmesine yardımcı olabilir. Bileşenlerin daha iyi bir şekilde belgelenmesi ve izlenmesi, organizasyonun sahip olduğu yazılım kaynaklarını daha iyi anlamasına yardımcı olabilir.

- Daha İyi Müşteri Hizmeti: Yeniden yapılandırma, müşteri hizmeti uygulamalarını geliştirmek için kullanılabilir. Müşteri geri bildirimlerine dayalı olarak mevcut sistemlerde iyileştirmeler yapmak, müşteri memnuniyetini artırabilir.

- Rekabet Avantajı: Yeniden yapılandırma, organizasyonun rakiplerine göre daha hızlı ve etkili bir şekilde yazılım tabanlı çözümler sunmasına yardımcı olabilir. Bu, organizasyonun rekabet avantajını artırabilir.

Bir işletmenin iş süreçlerinin yeniden yapılandırılması (Business Process Reengineering-BPR) çabalarını incelerken, bilgi sistemleri denetçisinin aşağıdakilerin olup olmadığını tespit etmesi gerekir:

- BPR ekibi, BPR/süreç değişikliği projesinin tamamlanmasından sonra öğrenilecek dersleri belgelemiştir. İşletmenin değişim çabaları, işletmenin genel kültürü ve stratejik planı ile tutarlıdır.

- Yeniden yapılanma ekibi, değişimin işletme personeli üzerinde yapabileceği olumsuz etkileri en aza indirmeye çalışmaktadır.

- Yeniden yapılandırma sürecinin, işletme içindeki tüm paydaşlara etkili bir şekilde iletilmesi ve anlaşılması için bir iletişim stratejisi olup olmadığını değerlendirmek önemlidir. İşletmenin çalışanları, müşterileri ve diğer paydaşları, değişimlerin nedenini ve sonuçlarını anlamalıdır.

- BPR projeleri her zaman belirli riskler taşır. Bilgi sistemleri denetçisi, proje ekibinin bu riskleri tanımlayıp yönetme stratejilerini değerlendirmelidir. Özellikle güvenlik, veri kaybı veya operasyonel kesintiler gibi riskler üzerinde durulmalıdır.

- BPR projelerinin uygulanabilirliği ve izlenmesi çok önemlidir. Bilgi sistemleri denetçisi, projenin gerçek dünyada nasıl işlediğini ve beklenen sonuçların elde edilip edilmediğini değerlendirmelidir.

- BPR projelerinin maliyet etkinliği ve performans ölçütleri göz önünde bulundurulmalıdır. Bu projelerin maliyetleri kontrol altında mı, beklenen getiri sağlanıyor mu ve süreçlerin verimliliği artırılıyor mu gibi sorular önemlidir.

- BPR projelerinin, geçerli yasal düzenlemelere ve endüstri standartlarına uygun olup olmadığı bilgi sistemleri denetçisi tarafından denetlenmelidir. Yasal sorumluluklar ve uyumluluk gereksinimleri dikkate alınmalıdır.

Tersine Mühendislik:

Tersine mühendislik, bilgisayar destekli yazılım mühendisliği (CASE) teknolojisi kullanılarak mevcut uygulama sistem kodunun yeniden tasarlanıp kodlanabildiği bir yazılım mühendisliği tekniğidir. Uygulamada tersine mühendisliğin iki çeşidi bulunur. İlki, yazılım için var olan kaynak kodu programda kötü şekilde yazılmış veya yazılıp da artık geçerli olmayan yüksek seviye görünüşlerinin bulunmasıdır. İkincisi, uygun kaynak kodu bulunmayan yazılım için kaynak kodunu bulmak için yapılan bütün girişimleri içerir. Tersine mühendislik, yazılım bakım/iyileştirilmesi kapsamında ilgili kaynak kodun anlaşılması, yazılım geliştirme sürecinde alınacak bir karar için gerekli bilgilerin çıkarılması, bir yazılım hatasının veya güvenlik açığının tespit edilmesi/düzeltilmesine yardımcı olabilir.

Aşağıda tersine mühendisliğin temel unsurlarına yer verilmiştir:

- Var olan Kodu Anlama: Tersine mühendislik, mevcut bir yazılımın kaynak kodunu analiz ederek çalışma prensiplerini ve yapısını anlamayı amaçlar. Bu, kodun yazarı veya ekip dışındaki kişilerin yazılımı incelemesini kolaylaştırır.

- Bilgi Çıkarma: Tersine mühendislik, yazılımın işlevselliği hakkında bilgi çıkarmayı amaçlar. Bu, yazılımın nasıl çalıştığını ve kullanıldığını anlamak için önemlidir. Ayrıca, yazılım hatalarını tespit etmek veya güvenlik açıklarını bulmak için de kullanılabilir.

- Yazılım İyileştirmesi: Mevcut bir yazılımın kodunu inceleyerek, iyileştirmeler veya güncellemeler eklemek mümkün olabilir. Bu, yazılımın performansını artırmak, hataları düzeltmek veya yeni özellikler eklemek için yapılabilir.

- Belgeleme: Tersine mühendislik, yazılımın dokümantasyonunu oluşturmayı veya güncellemeyi amaçlayabilir. Bu, yazılımın nasıl kullanılacağını, nasıl yapılandırılacağını ve nasıl bakım yapılacağını anlatan belgelerin oluşturulmasını içerebilir.

- Yeniden Tasarım ve Yeniden Kodlama: Tersine mühendislik, mevcut yazılımın yeniden tasarlanmasını ve yeniden kodlanmasını sağlayabilir. Bu, yazılımın eski veya karmaşık bir yapıya sahip olduğu durumlarda daha modern, verimli ve güncel bir kod tabanı oluşturmayı amaçlayabilir.

- Güvenlik İyileştirmesi: Tersine mühendislik, yazılımın güvenlik açıklarını tespit etmeye ve düzeltmeye yardımcı olabilir. Bu, potansiyel güvenlik tehditlerine karşı yazılımın daha savunmasız olmamasını sağlamak için yapılır.

- Karar Alma: Tersine mühendislik, yazılım geliştirme süreçlerinde alınacak kararlar için mevcut kodun analizine dayalı veri sağlayabilir. Bu, yazılımın gelecekteki gelişim yönünü belirlemeye yardımcı olabilir.

- Tersine mühendislik, yazılım geliştirme süreçlerinde ve yazılım bakımında önemli bir rol oynar. Mevcut yazılım sistemlerini anlamak ve iyileştirmek için kullanılan bu yöntem, yazılım endüstrisinde yaygın bir uygulamadır.

Bilgi sistemleri denetçisi aşağıdaki risk maddelerinden haberdar olmalıdır:

- Geri derleyiciler belirli bilgisayarlara işletim sistemlerine ve programlama dillerine bağlı işlevlere sahip araçlardır. Bu bileşenlerden birindeki herhangi bir değişiklik, yeni bir geri derleyici geliştirmeyi veya satın almayı gerektirebilir.

- Yazılım lisans sözleşmeleri genellikle lisans sahibinin yazılımı tersine mühendislik yapmasını saklayan hükümler içerir. Böylece hiçbir ticari sır ve programlama tekniği tehlikeye girmez.

DevOPS:

DevOps, çatışmaların ve engellerin ortadan kaldırılabilmesi için geliştirme ve operasyon süreçlerinin entegrasyonunu ifade eden yaşam döngüsüdür. Her bir aşama bir diğeriyle bağlantılıdır ve aşamalar role özel değildir. Gerçek bir DevOps kültüründe her rolün, aşamaların her biriyle belirli bir

ölçüde ilişkisi vardır. Bu entegrasyon büyük faydalar sağlayabilir ancak yeni riskler de yaratabilir. DevOps'u kullanma kararı bir kurumun ortamı, risk toleransı, kültürü ve geliştirme projesinin kapsamı gibi faktörlere dayanarak yapılmalıdır.

DevOps ortamı değiştirdiğinden ve genellikle bir işletmenin kontrol ortamını ve kabul edilen risk seviyesini etkilediğinden, bilgi sistemleri denetçisi görevlerin uygun şekilde ayrılmış (SoD) olduğundan emin olmalıdır.

DevOps Kullanmanın Önemi:

- Hızlı Dağıtım: DevOps, yazılımın hızlı ve sürekli olarak dağıtılmasını sağlar. Bu, yeni özelliklerin ve güncellemelerin daha hızlı bir şekilde kullanıcılara sunulabilmesi anlamına gelir.

- Daha İyi İşbirliği: Geliştirme (Development) ve Operasyon (Operations) ekipleri arasındaki iletişim ve işbirliği DevOps sayesinde artar. Bu, hataların ve sorunların daha hızlı çözülmesine olanak tanır.

- Otomasyon: DevOps, tekrarlayan görevlerin otomatikleştirilmesine odaklanır. Bu, iş süreçlerini verimli hale getirir ve insan hatalarını azaltır.

- İzlenebilirlik: DevOps süreçleri, yazılım geliştirme ve dağıtımının izlenebilirliğini artırır. Hangi kodun nerede ve ne zaman değiştirildiği, hata ayıklama ve sorun giderme süreçlerini kolaylaştırır.

- Yenilik ve Rekabetçilik: Hızlı dağıtım, yeni özelliklerin ve ürünlerin daha hızlı bir şekilde piyasaya sürülmesine olanak tanır. Bu da şirketlerin daha yenilikçi ve rekabetçi olmasına yardımcı olur.

DevOps'un Avantajları:

- Hızlı Teslimat: Yazılımın daha hızlı bir şekilde kullanıcılara sunulması, iş süreçlerinin hızını artırır. Bu, müşteri memnuniyetini artırabilir.

- Daha İyi Kalite: Otomasyon ve sürekli testler, yazılımın daha yüksek kalitede olmasını sağlar. Bu da hataların azalmasına ve daha güvenilir bir hizmet sunulmasına yardımcı olur.

- Düşük Maliyet: Otomasyon ve tekrar kullanılabilirlik, yazılım geliştirme maliyetlerini azaltabilir. Ayrıca hataların erken tespit edilmesi, maliyetli düzeltmeleri önler.

- Daha İyi İşbirliği: Ekipler arasındaki işbirliği ve iletişim, daha iyi fikir alışverişine ve sorunların daha hızlı çözülmesine yol açar.

- İş Sürekliliği: İş sürekliliği ve felaket kurtarma planlarının test edilmesi ve güncellenmesi, iş sürekliliğini sağlar.

- Ölçeklenebilirlik: DevOps, büyüyen ve değişen gereksinimlere uyum sağlama kapasitesine sahiptir. Sistemler daha kolay bir şekilde ölçeklenebilir.

- Güvenlik: DevOps genel güvenliği artırmaya yönelik birkaç farklı katman sağlar. İşlevsel açıdan, bazı durumlarda güvenlik ekiplerinin entegrasyonunu içerir. Bu, zaman zaman DevSecOps adlı bir model olarak ifade edilir. Bu model, geliştirme ve operasyon ihtiyaçlarının yanı sıra güvenliği eşit derecede dengeler. DevSecOps, güvenliği; sürekli entegrasyon, sürekli teslimat ve sürekli dağıtım hattına entegre ederek geliştirme sürecinin aktif, entegre bir parçası olarak rol oynar. Bu sayede iş akışı güvenli hale gelir.

- Rekabet Üstünlüğü: Hızlı dağıtım ve yenilik, şirketlerin rakiplerine göre daha hızlı hareket etmelerini ve pazarda öne geçmelerini sağlar.

DevOps'un iş süreçlerine getirdiği bu avantajlar, günümüzün hızla değişen iş dünyasında rekabetçi kalmak isteyen organizasyonlar için kritik öneme sahiptir. Bu nedenle birçok şirket, DevOps'u benimsemekte ve uygulamaktadır.

Bilgi sistemleri denetçileri, DevOps'un uygulanması sırasında aşağıdaki risk ve denetim noktalarını göz önünde bulundurmalıdır:

- Yetkilendirme ve Erişim Kontrolleri: DevOps ortamında, farklı ekipler ve roller arasında işbirliği artar. Bu nedenle, sistemlerin ve verilerin uygun bir şekilde korunduğundan emin olmak için

yetkilendirme ve erişim kontrolleri dikkatlice yönetilmelidir. Denetçiler, her rolün yalnızca gereksinim duyduğu sistemlere ve verilere erişebildiğini doğrulamalıdır.

- Değişiklik Yönetimi: DevOps süreçleri hızlı değişikliklere ve dağıtımlara izin verir. Bu, denetçilerin tüm değişikliklerin izlendiğinden, belgelenip test edildiğinden ve uygun onaylar alındığından emin olmalarını gerektirir. Ayrıca, değişikliklerin iş sürekliliği ve güvenlik gereksinimlerini karşıladığını doğrulamak da önemlidir.

- Güvenlik ve Sızma Testleri: DevOps süreçleri sırasında yazılımın hızlı bir şekilde geliştirilip dağıtılması, güvenlik açıklarının kaçırılma riskini artırabilir. Bilgi sistemleri denetçileri, güvenlik ve sızma testlerinin düzenli olarak gerçekleştirildiğini ve bulunan açıkların hızla düzeltildiğini kontrol etmelidir.

- Üçüncü Taraf Entegrasyonları: DevOps süreçlerinin bir parçası olarak üçüncü taraf hizmetleri ve bileşenleri entegre etmek yaygındır. Denetçiler, bu entegrasyonların iş sürekliliği, gizlilik ve uyumluluk gereksinimlerini karşıladığını doğrulamalıdır.

- İzlenebilirlik ve Günlüğe Alma: DevOps süreçleri, izlenebilirlik ve günlüğe alma önemli hale getirir. Denetçiler, tüm işlemlerin uygun bir şekilde günlüğe alındığını ve bu kayıtların gerektiğinde incelenebilir olduğunu kontrol etmelidir.

- İş Sürekliliği ve Felaket Kurtarma: Hızlı değişiklikler ve dağıtımlar, iş sürekliliği ve felaket kurtarma planlarının etkinliğini sınayabilir. Denetçiler, bu planların düzenli olarak test edildiğini ve güncellendiğini doğrulamalıdır.

- Kültürel ve Eğitimsel Uyum: DevOps, bir organizasyonun kültürünü ve çalışma şeklini etkileyebilir. Denetçiler, ekiplerin bu yeni yaklaşıma uyum sağlaması için gereken eğitim ve rehberliği aldığından emin olmalıdır.

Bilgi sistemleri denetçileri, DevOps'un getirdiği hız ve esneklik avantajlarını sürdürürken güvenlik, uyumluluk ve iş sürekliliği gereksinimlerinin karşılandığını doğrulamak için bu risk ve denetim noktalarına özellikle dikkat etmelidirler.

İş Süreçlerinin Yeniden Yapılandırılması ve Süreç Değişimi:

İş süreçlerinin yeniden yapılandırılması (BPR), mevcut iş ortamında hayatta kalabilmek için mevcut süreçlerin yeniden yapılandırılmasına dayanmaktadır. Bu süreç genelde manuel sistem süreçlerinin otomatizasyonuyla ilişkindir. Manuel müdahale ve kontrollerin azaltılması hedeflenir.

BPR adımlara aşağıda yer verilmektedir:

- Yeni Sürecin Uygulanması ve İzlenmesi: Yeniden yapılandırılan iş sürecinin uygulanması ilk adımdır. Bu adımda, yeni süreç, belirlenen tasarım ve düzenlemelere göre hayata geçirilir. Sürecin uygulanması sırasında performans ölçütleri ve anahtar göstergeler izlenir.

- Bir Proje Planının Geliştirilmesi: BPR projesinin başarılı bir şekilde yönetilebilmesi için bir proje planı geliştirilir. Bu plan, projenin hedeflerini, zaman çizelgesini, kaynakları ve sorumlulukları içerir. Proje planı, BPR sürecinin düzenli ve kontrol altında ilerlemesini sağlar.

- Sürekli Bir İyileştirme Sürecinin Oluşturulması: BPR, sadece mevcut süreçleri yeniden yapılandırmakla kalmaz, aynı zamanda sürekli bir iyileştirme kültürünün oluşturulmasına da katkı sağlar. Sürekli olarak iş süreçlerini izlemek, değerlendirmek ve geliştirmek, organizasyonun rekabetçiliğini artırır.

- Gözden Geçirilecek Alanların Tanımlanması: BPR projeleri, genellikle öncelikli olarak iyileştirilmesi gereken iş süreçlerini belirler. Bu adımda, organizasyonun hangi alanlarının yeniden yapılandırmaya ihtiyaç duyduğu net bir şekilde tanımlanır.

- İncelenmekte Olan Süreci Anlama: Bu aşamada, mevcut iş süreçleri derinlemesine incelenir ve anlaşılır. Sürecin her aşaması, katılımcılar ve kaynaklar dikkate alınarak ayrıntılı olarak analiz edilir.

- Sürecin Yeniden Tasarım ve Düzenlenmesi: En kritik adım, mevcut süreçlerin yeniden tasarımıdır. Bu aşamada, belirlenen zayıf noktaları ele almak, verimliliği artırmak ve hedeflenen sonuçları elde etmek için süreçlerin nasıl yeniden şekillendirileceği belirlenir.

BPR'de kilit kontroller, iş süreci dışında yeniden yapılandırılabilmektedir. Bu yapılandırma bilgi sistemleri denetçisi tarafından özellikle denetlenmelidir. BPR projeleri, iş süreçlerini temelinden değiştirebileceği için bilgi sistemleri denetçisi için önemlidir. Aşağıda, bu önemi açıklayan maddelere yer verilmektedir:

- İç Kontrollerin Yeniden Değerlendirilmesi: BPR süreci, iş süreçlerinin temel değişikliklerini içerdiği için iç kontrollerin de gözden geçirilmesini gerektirir. Bilgi sistemleri denetçisi, yeni süreçlerin ve iş akışlarının güvenlik ve uyumluluk açısından nasıl etkilendiğini değerlendirmelidir.

- Veri Yönetimi ve Entegrasyonu: BPR, genellikle farklı sistemlerin ve uygulamaların entegrasyonunu içerir. Bilgi sistemleri denetçisi, verilerin doğru bir şekilde yönetilmesini ve farklı sistemler arasında sorunsuz bir veri akışını sağlamak için önemli bir rol oynar.

- Yazılım ve Uygulama Değişiklikleri: BPR projeleri, yeni yazılımların ve uygulamaların geliştirilmesini veya mevcut olanların değiştirilmesini içerebilir. Bu değişiklikler, iş süreçlerinin daha iyi desteklenmesi için yapılırken, bilgi sistemleri denetçisi bu yazılımların güvenlik ve uyumluluk gereksinimlerini karşılayıp karşılamadığını değerlendirmelidir.

- Eğitim ve Farkındalık: BPR projeleri genellikle organizasyon içinde büyük değişiklikler getirir. Bilgi sistemleri denetçisi, çalışanların bu değişikliklere uyum sağlaması için eğitim ve farkındalık programlarının etkili bir şekilde uygulanmasını izlemelidir.

- Sürekli Denetim ve İzleme: BPR sonrası süreçlerin etkinliği ve uygunluğu düzenli olarak izlenmelidir. Bilgi sistemleri denetçisi, bu izleme süreçlerini oluşturmalı ve sürekli denetimleri gerçekleştirmelidir.

Bu nedenlerle, BPR projeleri bilgi sistemleri denetçisi için kritik bir öneme sahiptir çünkü iş süreçlerinin yeniden yapılandırılması organizasyonun bütününü ve bilgi sistemlerini derinden etkileyebilir.

2.1.4. Sistem Geliştirme Araçları ve Kod Geliştirme Yardımcıları

Sistem geliştirme sürecinde muhtelif araç ve kod geliştirme yardımcıları kullanılmaktadır. Bu kısımda, söz konusu araç ve yardımcılara yer verilecektir.

Bilgisayar Destekli Yazılım Mühendisliği (Computer-aided Software Engineering-CASE):

Bilgisayar destekli yazılım mühendisliği (CASE), yazılım geliştirme sürecinde yüksek kaliteli, hatasız ve bakımı kolay yazılım ürünleri sağlamak için bilgisayar destekli yazılım araç ve yöntemlerin yazılım geliştirilmesinde bilimsel uygulamasıdır. CASE, kontrol odaklı ve disiplinli bir yaklaşım sağlar. Tasarımcılar, geliştiriciler, testçiler, yöneticiler ve diğerlerinin geliştirme sırasında proje aşamalarını görmelerine yardımcı olur. Bilgi sistemleri denetçisi, CASE'in getirdiği gelişim surecindeki değişiklikleri tanıyabilmeli ve CASE'i bir denetim aracı olarak kullanabilmelidir.

Yazılım sistemleri, yazılım süreç aktivitelerine otomatik destek sağlamaya yöneliktir. CASE sistemleri metot desteği için kullanılır.

CASE, uygulamaları tasarlamak ve uygulamak için kullanılan yazılım araçları etki alanıdır. CASE araçları bilgisayar destekli tasarım, donanım ürünleri tasarlamak için kullanılan (CAD) araçlarına benzerler. CASE yazılımı, genellikle yazılım geliştirme sürecinde kullanılabilecek otomatikleştirilmiş araçlarla birlikte bilgi sistemlerinin geliştirilmesi için yöntemlerle ilişkilendirilir.

CASE yaklaşımının avantajları aşağıdakileri içerir:

- Servis Maliyetlerinde Azalma: Testin yanı sıra yeniden tasarlamaya da önem verildiğinden, bir ürünün beklenen ömrü boyunca servis maliyeti önemli ölçüde azalır.

- Ürün Kalitesinde Artış: Geliştirme sürecinde organize bir yaklaşım izlendiğinden ürünün genel kalitesi artar.

- Gereksinimlerin Daha İyi Karşılanması: Müşterilerin sürecin bir parçası olarak kalmasını sağladığı için gerçek dünya gereksinimlerini karşılama şansı daha olası ve daha kolaydır.

- Rekabet Avantajı Sağlama: Dolaylı olarak, yüksek kaliteli ürünlerin geliştirilmesine yardımcı olarak işletmeye rekabet avantajı sağlar.

- Daha Hızlı Geliştirme Süreci: CASE araçları, yazılım geliştirme sürecini otomatikleştirerek ve standartlaştırarak zaman tasarrufu sağlar. Bu, yeni projelerin daha hızlı başlatılmasını ve yazılımın daha hızlı bir şekilde pazara sunulmasını mümkün kılar.

- Verimli Kaynak Kullanımı: CASE, yazılım geliştirme kaynaklarının daha etkili bir şekilde kullanılmasına yardımcı olur. Geliştiriciler, daha az zaman harcayarak daha fazla iş başarabilirler, bu da maliyetleri azaltır.

- Daha İyi İşbirliği: CASE araçları, geliştirme ekibinin işbirliği yapmasını kolaylaştırır. Proje paydaşları, aynı platformda çalışabilir, iletişim ve işbirliği daha etkili hale gelir.

- Daha İyi Belgeleme: CASE araçları, yazılım projeleri için otomatik belge oluşturma yeteneği sağlar. Bu, yazılımın daha iyi belgelenmesine ve sürdürülebilirliğinin artmasına katkıda bulunur.

- Hata Azaltma: Otomatik analiz ve kod oluşturma yetenekleri sayesinde CASE araçları, hataların erken tespit edilmesine ve düzeltilmesine yardımcı olur. Bu da yazılımın daha güvenilir olmasını sağlar.

- Daha İleri Analiz: CASE araçları, daha karmaşık analizler yapma yeteneği sunar. Büyük veri işleme, veri madenciliği ve yapay zeka gibi gelişmiş analitik işlevleri entegre etmek daha kolaydır.

- İş Sürekliliği: CASE araçları, iş süreçlerini ve uygulamaları belirli bir düzen ve süreklilik içinde sürdürmeye yardımcı olur. Bu, iş sürekliliği planlaması ve felaket kurtarma için önemlidir.

CASE yaklaşımının dezavantajları aşağıdakileri içerir:

- Maliyet: Küçük ölçekli işletmeler CASE araçlarına yeterli yatırımı yapamazlar.

- Öğrenme: İlk aşamalarda teknoloji öğrenildiği için verimlilik düşük olur.

- Organize Araç: Maliyet avantajı sağlamak için uygun bir araç seçimi karışımı oluşturmak önemlidir.

CASE araçları kaliteli, kusur içermeyen, kullanışlı ve sürdürülebilir yazılımları geliştirmek, yazmak, kodlamak ve tasarlamak için kullanılır. Burada amaç, yazılım geliştirme süreçlerinin kontrol edilebilmesi, ölçeklenebilmesi ve kolay yönetilebilmesidir. CASE araçları, kriterlerine ve türlerine göre aşağıda yer verilen gruplarda incelenebilir:

- Kod Üreten Yazılım Araçları Grubu: Bu grup, belirli bir yazılım projesi için programlama kodlarını otomatik olarak üreten araçları içerir. Bu araçlar, geliştiricilere kod yazma işlemine yardımcı olur ve hızlandırır. Örneğin, belirli bir programın temel yapısını oluşturan kod bloklarını üretebilirler. Bu, yazılım geliştirme sürecini hızlandırır ve insan hatalarını azaltır.

- Tasarım Araçları Grubu: Tasarım araçları, yazılım projelerinin tasarım aşamasında kullanılır. Bu araçlar, veritabanı şemaları, kullanıcı arayüzleri, sistem mimarileri ve diğer tasarım belgelerini oluşturmak için kullanılabilir. Bu sayede geliştiriciler, projenin nasıl görüneceği ve nasıl çalışacağı konusunda net bir vizyon geliştirebilirler.

- Veri Modelleme Araçları Grubu: Bu grup, veritabanı tasarımı ve yönetimi için kullanılır. Veri modelleme araçları, veritabanlarının yapısını oluşturmak, ilişkileri tanımlamak ve veri akışını yönetmek için kullanılır. Bu, veritabanı işlemlerini optimize etmek ve veri bütünlüğünü korumak için önemlidir.

- Versiyon Kontrol ve Ayarlama Araçları Grubu: Bu araçlar, yazılım geliştirme sürecini izlemek, sürümleri yönetmek ve kod değişikliklerini takip etmek için kullanılır. Geliştiricilerin eşgüdüm içinde çalışmalarına ve kod tabanını güncellemelerine yardımcı olur. Aynı zamanda yazılımın farklı sürümlerini yönetmek için kullanılır.

- Model Dönüşüm Yazılımları Grubu: Model dönüşüm araçları, farklı yazılım modelleme dilleri veya platformları arasında dönüşüm yapmayı sağlar. Örneğin, bir modeli başka bir programlama diline veya platforma dönüştürebilirler. Bu, yazılım geliştirme sürecinde farklı gereksinimlere uyum sağlamak için önemlidir.

- Kod Yeniden Üretim Yazılımları Grubu: Bu araçlar, mevcut yazılım kodunu otomatik olarak analiz eder ve kodun kalitesini artırmak veya belirli özellikleri eklemek için kodu değiştirmenizi sağlar. Aynı zamanda kodun daha temiz ve anlaşılır olmasını sağlarlar.

CASE (Computer-Aided Software Engineering) Araçlarının Modern Teknolojilerle Entegrasyonu ve Verimlilik Artışı

CASE (Computer-Aided Software Engineering) araçları, yazılım geliştirme sürecinin her aşamasında önemli bir rol oynar ve bu araçlar yazılımın kalitesini artırırken geliştirme sürecini hızlandırır. Bu araçlar, yazılım tasarımından uygulamaya alma aşamasına kadar olan tüm süreçleri destekler. CASE araçlarının temel amacı, yazılım geliştirme süreçlerini daha verimli, hatasız ve sürdürülebilir hale getirmektir. Yazılım mühendisliğinde kullanılan CASE araçları, genellikle otomatikleştirilmiş süreçleri içerir ve bu araçlar, yazılım projelerinin planlamasından bakım aşamasına kadar her adımda kullanılabilir. Aşağıda, CASE araçlarının çeşitli grupları ve bu araçların yazılım geliştirme sürecine nasıl katkı sağladığına dair detaylı bir açıklama bulunmaktadır:

CASE Araçlarının Uygulama Alanları ve Kullanımı: CASE (Computer-Aided Software Engineering) araçları yazılım geliştirme sürecini hızlandırır, ancak aynı zamanda yazılımın kalitesini artırmak, hataları erken tespit etmek ve daha verimli işbirliği sağlamak için de kritik öneme sahiptir. Bu araçlar, özellikle büyük çaplı projelerde zaman tasarrufu sağlar ve küçük işletmelerin yazılım geliştirme süreçlerini daha verimli hale getirir. CASE araçları, yazılım projelerinin her aşamasında kullanılabilir ve çeşitli yazılım geliştirme metodolojileriyle uyumlu çalışabilir.

- Büyük ve küçük işletmelerde CASE araçlarının rolü: CASE araçları, yalnızca büyük yazılım geliştirme projelerinde değil, küçük işletmelerde de yazılım geliştirmeyi hızlandırmak ve standartlaştırmak için kullanılabilir.

- Projelerde zaman tasarrufu ve işbirliği: Yazılım geliştirme süreçlerinde, özellikle takım çalışması gerektiren büyük projelerde, CASE araçları zaman tasarrufu sağlar ve proje ekiplerinin daha verimli çalışmasını sağlar.

CASE Araçlarının Yeni Nesil Teknolojilerle Uyumluluğu: CASE araçları, yeni nesil yazılım geliştirme metodolojileri ve teknolojileri ile uyumlu hale gelmiştir. Örneğin, yapay zeka (AI), veri analitiği ve bulut tabanlı yazılım geliştirme gibi yenilikçi teknolojilerle entegre edilmesi, yazılım geliştirme süreçlerini daha hızlı ve verimli hale getirir. Bu tür entegrasyonlar, yazılımın kalitesini artırır ve geliştirilmekte olan yazılımların sürdürülebilirliğini sağlar.

- Yapay zeka ve CASE entegrasyonu: Yapay zeka ile desteklenen CASE araçları, yazılım geliştirme sürecinde otomatik hata düzeltmeleri ve öneriler sunar, bu da geliştirme sürecini hızlandırır.

- Veri analitiği ve CASE araçları: CASE araçları, büyük veri analitiği ile entegre çalışarak, yazılım projelerinde gelişmiş analizler yapma ve verimlilik sağlama imkanı sunar.

- Bulut tabanlı CASE araçları: Bulut teknolojileri ile entegre olan CASE araçları, uzak ekiplerin de projelere katkı sağlamasını kolaylaştırarak global işbirliğini artırır.

CASE Araçlarının Entegre Edilmesi ve Etkin Kullanımı: CASE araçları, yazılım geliştirme sürecinin farklı aşamalarında yer alır ve diğer yazılım araçlarıyla entegrasyon yaparak projelerin daha verimli olmasını sağlar. Özellikle, yazılım geliştirme sürecindeki farklı araçların entegre çalışabilmesi, daha tutarlı bir yazılımın geliştirilmesine katkıda bulunur. Bu sayede, projelerdeki veriler farklı sistemler arasında kolayca paylaşılarak bilgi kaybı yaşanmaz.

- Araç entegrasyonu ve veri paylaşımı: Farklı yazılım geliştirme araçları arasında entegrasyon yapılarak, verilerin doğru ve güncel bir şekilde paylaşılması sağlanır.

- Yazılım araçlarının uyumluluğu: CASE araçları, yazılım geliştirme sürecinin başlangıcından sonuna kadar entegre çalışarak veri kaybını önler ve projelerdeki hataları minimuma indirir.

CASE Araçlarının Otomasyon ve Verimlilik Sağlamadaki Rolü: CASE araçları, yazılım geliştirme sürecinde otomasyonu artırarak zaman kaybını azaltır ve kaynakların daha verimli kullanılmasını sağlar. Özellikle yazılım testleri, kod analizleri ve hata ayıklama gibi rutin işlemler otomatikleştirilebilir. Bu otomasyon, hem geliştiricilerin daha hızlı çalışmasına olanak tanır hem de yazılımın kalitesini artırır.

- Testlerin otomatikleştirilmesi: CASE araçları, yazılım testlerini otomatikleştirerek geliştirme sürecini hızlandırır ve hataları erken aşamalarda tespit eder.

- Kod analizlerinin otomatikleştirilmesi: Kod kalitesinin sürekli olarak izlenmesi, hataların erkenden tespit edilmesini sağlar ve yazılımın daha stabil olmasına katkı sağlar.

Yeni Nesil CASE Araçları ve Bulut Teknolojileri: Yeni nesil CASE araçları, bulut tabanlı çözümler ile entegre edilerek daha esnek, ölçeklenebilir ve erişilebilir hale gelmiştir. Bulut teknolojileri sayesinde yazılım geliştirme süreci, dünya çapındaki ekipler tarafından daha verimli bir şekilde yürütülür. Bu entegrasyon, projelerin hızlıca başlatılmasını ve pazara sunulmasını sağlar.

- Bulut tabanlı yazılım geliştirme: Bulut teknolojileri ile entegre CASE araçları, yazılım projelerinin yönetilmesini daha verimli hale getirir ve dünya çapında işbirliği yapılmasını sağlar.

- Veri güvenliği ve bulut teknolojileri: Bulut tabanlı CASE araçları, veri güvenliğine özel önlemler alarak güvenli yazılım geliştirme süreçleri sağlar.

CASE Araçları ve Süreç İyileştirmeleri: CASE araçları, yazılım geliştirme süreçlerini optimize eder ve süreçlerdeki verimliliği artırır. Bu araçlar, yazılım projelerinin her aşamasında kesintisiz bir iş akışı sağlar ve süreçlerdeki hataları azaltır. Böylece, yazılım geliştirme süreci daha hızlı ve güvenilir hale gelir.

- Sürekli iyileştirme ve CASE araçları: Yazılım geliştirme sürecinde sürekli iyileştirme sağlayan CASE araçları, süreçlerin daha hızlı ve verimli yapılmasını sağlar.

- Geliştirme sürekliliği ve güvenilirlik: CASE araçları sayesinde yazılım projelerinin geliştirilmesi daha sürdürülebilir ve güvenilir hale gelir.

Modern Kavramlar: Yeni nesil yazılım geliştirme araçları, sadece yazılım üretmekle kalmaz, aynı zamanda yazılım geliştirme sürecinde işbirliği, veri analizi ve güvenlik gibi alanlarda da iyileştirmeler sağlar. Bu araçların entegre bir şekilde çalışması, modern yazılım projelerinde başarıyı garantileyen kritik faktörlerden biridir.

- Yapay zeka ve CASE araçları: Yapay zeka destekli CASE araçları, yazılım geliştirme sürecinde otomatik hata tespiti, analiz ve geliştirme önerileri sunarak yazılım projelerinin başarısını artırır.

- Veri analitiği ve işbirliği: CASE araçları, büyük veri analitiği ile entegre çalışarak, yazılım geliştirme sürecinde elde edilen verilerin daha verimli bir şekilde kullanılmasına olanak tanır.

Kod Üreteçleri:

Yazılım dünyasındaki nesne tabanlı dillerin gelişmesiyle programcılık, yazılım algoritmaları, veri tabanı kavramları ve yazılım süreçleri değişimlere uğramıştır. Bir bilgi sistemleri denetçisi, sistem analisti tarafından tanımlanan parametrelere veya bir CASE ürününün tasarım modülü tarafından geliştirilen veri/varlık akış şemalarına dayanarak program kodu üreten araçlar ve bu tür araçlar tarafından üretilen kaynak kodunun farkında olmalıdır.

Kod üreteçleri, temel olarak kullanıcının ne yapmak istediğini programa öğretmesiyle başlar. Genelde kullanıcının istediği giriş forumları, kayıt girme, düzeltme, silme gibi bir temel işlemlerdir. Eğer mantıksal işlemlere yapacağı zaman ise yine kod yazma araçları bazı işlemleri gerçekleştirmektedir.

Bu değişimlerin en önemli süreci makro programlama olarak nitelendirilen "*yapacağın işlemleri bana söyle ben yapayım*" programlama türüdür. Başka bir tanımı ise paket programların bazılarının içerisinde bulunan, kullanıcıya kolaylık olması açısından, sürekli tekrar edilen (rutin) işlemlerin otomatik hale getirilmesi için kullanılan bir komut/komutlar dizisidir. Çünkü bilinen bir şey var kullanıcı

bir nesneyi seçer ve ona değişik hareketler kazandırır. Yazma, silme, yazdırma, seçme, hareket etme gibi birçok özellik kazandırabilir. Bunu yaparken hiçbir programlama da bilmez. Program kendi kodunu kendisi yazar.

Bilgi sistemleri denetçileri için bu araçların nasıl çalıştığını ve kullanıldığını anlamak önemlidir. Aşağıda kod üreteçleri hakkında daha fazla bilgiye yer verilmiştir:

- Başlangıç Noktası: Kod üreteçleri, kullanıcıların yazılım için temel gereksinimleri belirledikleri bir başlangıç noktasıyla çalışır. Kullanıcılar, genellikle kullanıcı arabirimi veya tasarım modelleri aracılığıyla istedikleri işlevselliği tanımlarlar.

- Temel İşlemler: Kod üreteçleri, temel işlemleri otomatik olarak oluşturmak için kullanılır. Bu işlemler, kullanıcının belirttiği işlevselliklerin temel yapı taşlarıdır. Örneğin, kullanıcıların veri girişi, kayıt, düzenleme, silme gibi temel işlemleri tanımlamasına yardımcı olurlar.

- Mantıksal İşlemler: Bazı kod üreteçleri, kullanıcıların mantıksal işlemleri tanımlamalarına da olanak tanır. Bu, yazılımın daha karmaşık işlevselliği için gereklidir. Örneğin, bir işlem sırasında verilerin nasıl işleneceği veya koşullara bağlı olarak hangi adımların izleneceği gibi işlemleri belirlemeye yardımcı olabilirler.

- Makro Programlama: Bu tür araçlar, kullanıcıların "*yapacağın işlemleri bana söyle ben yapayım*" yaklaşımını benimser. Kullanıcı, belirli bir işlem veya işlevi tanımlar ve kod üreteci, bu tanımı temel alarak ilgili program kodunu otomatik olarak oluşturur. Bu, kullanıcıların derinlemesine programlama bilgisine ihtiyaç duymadan karmaşık işlevselliği uygulamalarına olanak tanır.

- Rutin İşlemlerin Otomatikleştirilmesi: Kod üreteçleri, sıkça tekrarlanan veya rutin olarak gerçekleştirilen işlemleri otomatik hale getirmek için kullanılır. Bu, yazılım geliştirme sürecini hızlandırır ve hata olasılığını azaltır.

- Kullanıcı Dostu Arayüzler: Kullanıcılar, genellikle kod üreteçleri aracılığıyla kullanıcı dostu arayüzler oluşturabilirler. Bu arayüzler, uygulamanın nasıl çalıştığını ve kullanıcıların nasıl etkileşimde bulunabileceğini görsel olarak tanımlar.

Kod üreteçleri, yazılım geliştirme sürecini daha verimli hale getirir ve geliştiricilerin daha fazla zamanlarını karmaşık işlevselliği tasarlamaya ve özelleştirmeye ayırmalarına yardımcı olur. Bu nedenle, bilgi sistemleri denetçileri için kod üreteçlerinin nasıl çalıştığını ve nasıl kullanıldığını anlamak önemlidir, çünkü bu araçlarla geliştirilen yazılımın denetimi de söz konusu olabilir.

Kod üreteçleri, yazılım geliştirme sürecinde devrim niteliğinde araçlar haline gelmiştir. Günümüzde yazılım geliştirme yöntemlerinin hızla evrilmesiyle birlikte, kod üreteçlerinin kullanımı daha da yaygınlaşmıştır. Bu araçlar yalnızca verimlilik sağlamakla kalmaz, aynı zamanda hata oranlarını da önemli ölçüde azaltır. Kod üreteçlerinin sunduğu avantajlar şu şekildedir:

- Hızlı Prototip Geliştirme: Kod üreteçleri, yazılım geliştirme sürecinin erken aşamalarında hızlı prototipler oluşturulmasına olanak tanır. Bu, özellikle kullanıcı geri bildirimlerini hızlı bir şekilde almak isteyen projelerde büyük avantaj sağlar. Kullanıcılar, sistemin işlevselliğini görmek ve test etmek için geliştirme sürecini hızlandırabilirler.

- Düşük Hata Oranı: Kullanıcılar, temel işlevleri kod üreteçleri aracılığıyla tanımlarken, yazılımın otomatik olarak üretilen kodu hatasız bir şekilde gerçekleştirmesini sağlar. Bu, manuel kod yazarken karşılaşılan yaygın yazılım hatalarının önlenmesine yardımcı olur. Ayrıca, otomatikleştirilmiş sistemlerin doğrulama ve test süreçlerinde daha az hata olasılığı bulunur.

- Esneklik ve Uyumluluk: Kod üreteçleri, kullanıcının gereksinimlerine göre özelleştirilebilir. Sistem geliştikçe, bu araçların sağladığı esneklik sayesinde, kullanıcılar işlevselliği değiştirebilir ve yeni özellikler ekleyebilirler. Bu, çevik yazılım geliştirme (Agile) metodolojisinin benimsenmesiyle de uyumludur, çünkü projelerin her aşamasında esnek değişiklikler yapılmasına olanak tanır.

- Yazılımın Modülerliği: Kod üreteçleri, yazılımın daha modüler bir yapıda geliştirilmesini sağlar. Bu modüler yapılar, yazılımın daha kolay bakımını ve gelecekteki güncellemeleri mümkün kılar. Modülerlik, ayrıca yazılımın çeşitli bölümlerinin bağımsız olarak güncellenmesini ve geliştirilmesini sağlar.

- Daha Az Teknik Borç: Kod üreteçleri, geliştirilen yazılımın temel işlevlerini ve modüllerini doğru ve verimli bir şekilde oluşturur. Bu, yazılımın teknik borç biriktirmesini engeller. Teknik borç, yazılımın gelecekteki geliştirmelerinin zorlaşmasına yol açabilecek kötü kodlama uygulamalarıdır.

- Veri Tabanı Entegrasyonu: Kod üreteçleri, veritabanlarıyla entegre çalışarak veri giriş, silme ve düzenleme işlemlerini kolaylaştırır. Bu, geliştiricilerin veri tabanı yönetim sistemi (DBMS) üzerinde çalışırken zaman kazanmasına ve sistemle ilgili olası hataları önceden fark etmelerine yardımcı olur.

Günümüzde Kod Üreteçlerinin Yeri ve Yazılım Geliştirme Süreçleri Üzerindeki Etkileri

Kod üreteçlerinin yazılım geliştirme sürecinde giderek daha önemli hale gelmesi, günümüzde yazılım projelerinin hızla evrimleşmesine olanak sağlamaktadır. Modern yazılım geliştirme çevrelerinde, kod üreteçleri yalnızca geliştiriciler için verimlilik artırıcı araçlar olmakla kalmaz, aynı zamanda tüm yazılım yaşam döngüsünü iyileştiren temel bileşenler olarak öne çıkar.

- Sürekli Entegrasyon ve Sürekli Dağıtım (CI/CD): Kod üreteçleri, sürekli entegrasyon ve sürekli dağıtım (CI/CD) süreçlerini destekler. Bu araçlar sayesinde geliştirilen yazılımın daha hızlı bir şekilde test edilip dağıtılmasına olanak sağlanır. Yazılımın her aşamasında yapılan değişiklikler hızlıca test edilip, hatalar en kısa sürede tespit edilir.

- Yapay Zeka Destekli Kod Üretimi: Günümüzde yapay zeka (AI) destekli araçlar, kod yazmayı daha da hızlandırmış ve geliştiricilerin karmaşık algoritmaları daha hızlı bir şekilde entegre etmelerini sağlamıştır. Bu araçlar, yazılımın çalışma mantığını anlayarak, kodu otomatik bir şekilde oluşturabilir ve geliştirme sürecini daha verimli hale getirebilir.

- Modüler Yazılım Geliştirme: Kod üreteçleri, yazılımın modüler bir yapıda inşa edilmesini sağlar. Bu, yazılımın farklı bileşenlerinin bağımsız olarak geliştirilmesi ve test edilmesi anlamına gelir. Modüler yapı, yazılım geliştirme sürecinde esneklik sağlar ve yazılımın gelecekteki güncellemelerini daha kolay hale getirir.

Kod Üreteçlerinin Geleceği

Kod üreteçlerinin yazılım geliştirme sürecindeki rolü, ilerleyen yıllarda daha da önemli hale gelecektir. Yapay zeka ve makine öğrenmesi gibi teknolojiler, bu araçları daha da güçlü hale getirecek ve yazılım geliştirme sürecini daha da hızlandıracaktır.

- Otomatik Kod Üretimi ve Test: Gelecekte kod üreteçleri, yalnızca kod üretmekle kalmayacak, aynı zamanda yazılımın tüm testlerini de otomatik olarak gerçekleştirecektir. Bu, yazılım geliştirme süreçlerini hızlandıracak ve hata oranlarını minimize edecektir.

- Yapay Zeka Destekli Kod Analizi: Kod üreteçlerinin geleceği, AI ile daha da zenginleşecektir. Bu araçlar, geliştirilen kodu analiz ederek potansiyel hataları önceden tespit edebilecek ve yazılımın performansını optimize edebilecektir.

- Daha Yüksek Uyumluluk: Kod üreteçlerinin geleceği, daha geniş yazılım sistemleri ile uyumlu olmayı amaçlayan araçlarla şekillenecektir. Yazılım geliştirme sürecinde kullanılan araçlar, diğer yazılım bileşenleriyle daha etkili bir şekilde entegre olacaktır.

Dördüncü Nesil Diller (Fourth-generation Languages-4**GLs):**

Dördüncü nesil (programlama) dil (4GL), 3GL'lerden insan diline, düşünme biçimine ve kavramsallaştırmaya yaklaşmaya çalışan bir grup programlama dilidir. 4. nesil dil, etki alanına özgü dil veya yüksek üretkenlik dili olarak da bilinir. Bu tür bir dilin amacı, profesyonel olmayan geliştiricilerin (örneğin, bilgisayar mühendisleri olmayanlar) kendi uygulamalarını geliştirmelerine izin vermektir. 4GL'ler, kullanımı çok daha kolay, daha az kod yazarak yönergeler, hazır şablonlar ve sihirbazlar sayesinde belirli ihtiyaçlarda uzmanlaşmış pratik çözümler geliştirmeye yönelik olup, yazılım geliştirmenin toplam süresi, çabası ve maliyetinin azaltmak için tasarlanmıştır.

4GL'lerin temel alanları ve ailelerini; veritabanı sorguları, rapor üreteçleri, veri işleme, analiz ve raporlama, ekran ressamları ve üreteçleri, GUI yaratıcıları, matematiksel optimizasyon, web geliştirme ve genel amaçlı dilleri oluşturmaktadır.

4GL'ler ortamsal açıdan bağımsız ve basit bir dil altkümesine ve bir tezgâh yaklaşımına sahip olan, prosedürel olmayan dillerdir. Bu tür diller, programcıların çok az kod içeren programlar yazmalarına izin verir ve bu da üretkenliklerini artırır. Bu dillerde yazılan programlar daha yavaş çalışır, daha fazla kaynak gerektirir ve programcılar bazı durumlarda 3 E veya 2 E nesli bir dil kullanmayı tercih ederler.

4GL'ler, yazılım geliştirme süreçlerini daha verimli ve anlaşılır hale getirmeyi amaçlayan programlama dilleridir. 4GL'lerin özellikleri ve avantajları aşağıda ele alınmıştır:

- İnsan Diline Yakın Olma: 4GL'ler, geliştiricilerin programlama sürecini daha anlaşılır ve insan diline yakın bir şekilde yapmalarına olanak tanır. Bu, karmaşık programlama kodlarını yazma ihtiyacını azaltır ve yazılım geliştirme sürecini daha erişilebilir hale getirir.

- Kavramsallaştırma ve Düşünme Biçimi: 4GL'ler, problemleri daha kavramsal bir şekilde ele almaya ve düşünme biçimine yaklaşmaya çalışır. Bu, geliştiricilerin uygulamalarını daha iyi tasarlamalarına yardımcı olur.

- Etki Alanına Özgü Dil: 4GL'ler, belirli bir iş veya etki alanı için optimize edilmiş dillerdir. Bu, geliştiricilerin özel gereksinimlere uygun uygulamalar oluşturmalarına olanak tanır.

- Yüksek Üretkenlik: Bu diller, yazılım geliştirme sürecini hızlandırmayı ve kodlama süresini kısaltmayı amaçlar. Böylece daha hızlı sonuçlar elde edilir.

- Veritabanı İşlemleri ve Raporlama: 4GL'ler, veritabanı sorguları oluşturmak, veri işlemek ve raporlar üretmek için kullanılır. Bu, veri yönetimi işlemlerini kolaylaştırır.

- Grafik Arayüzler ve GUI Geliştirme: 4GL'ler, grafiksel kullanıcı arayüzleri (GUI) oluşturmak ve kullanıcı dostu uygulamalar geliştirmek için kullanılır.

- Genel Amaçlı Diller: Bazı 4GL'ler, genel amaçlı programlama gereksinimlerini karşılamak için kullanılabilir, ancak bu genellikle daha uzun kodlar gerektirir.

- Üretkenlik Artışı: 4GL'ler, daha az kod yazma ihtiyacı nedeniyle geliştiricilerin daha hızlı çalışmalarına yardımcı olur. Bu, yazılım projelerinin süresini kısaltır.

Ancak, 4GL'lerin dezavantajları da vardır. Bunlara aşağıda yer verilmektedir:

- Daha yavaş çalışabilirler.

- Daha fazla kaynak gerektirebilirler.

- Bazı durumlarda 3GL veya 2GL gibi daha düşük seviyeli dillerin kullanılmasını gerektirebilirler.

- Belirli bir 4GL'nin öğrenilmesi ve kullanılması, geleneksel programlama dillerine göre daha fazla eğitim gerektirebilir.

Dördüncü Nesil Diller (4GL) ve Modern Uygulamalar

Dördüncü nesil diller (4GL'ler), yazılım geliştirmede devrim yaratmak için tasarlanmış, yazılım geliştirme sürecini hızlandıran, kullanıcı dostu ve üretkenliği artıran programlama dilleridir. 4GL'ler, genellikle veritabanı yönetimi, GUI (grafiksel kullanıcı arayüzü) tasarımı ve veri işleme alanlarında kullanılır. Bu diller, daha az kod yazmayı gerektirerek yazılım geliştirme sürecini daha hızlı ve verimli hale getirir. Günümüzde, 4GL'ler, modern yazılım geliştirme süreçlerinde büyük rol oynamaktadır. 4GL'ler hakkında bazı önemli hususlar aşağıda maddeler halinde ele alınmıştır:

Veri Madenciliği ve Yapay Zeka ile Entegrasyon: Modern 4GL'ler, veri madenciliği ve yapay zeka gibi ileri düzey teknolojilere kolayca entegre edilebilir. Özellikle büyük veri analitiği ve yapay zeka projelerinde, 4GL'ler yazılım geliştirmeyi hızlandırmak için kullanılır. Bu dillerin sağladığı hızlı veri işleme ve analiz kapasitesi, veriyi anlamak ve modellemek için büyük avantaj sağlar. Bu sayede yazılım geliştirme sürecindeki veri odaklı kararlar daha etkili hale gelir.

Bulut Tabanlı Uygulamalar: Günümüzde, 4GL'ler özellikle bulut tabanlı yazılım geliştirme süreçlerinde yoğun bir şekilde kullanılmaktadır. Bulut ortamlarında, 4GL'ler veri tabanı işlemleri,

sunucu tarafı işlemler ve uygulama entegrasyonları gibi görevleri hızlıca yerine getirebilir. Bu, kullanıcıların daha esnek ve ölçeklenebilir uygulamalar geliştirmelerini sağlar.

Mobil Uygulama Geliştirme: 4GL'ler, mobil uygulama geliştirme süreçlerinde de önemli bir yer tutmaktadır. Özellikle düşük kod (low-code) platformlarında kullanılan 4GL'ler, mobil uygulamaların daha hızlı ve kolay bir şekilde geliştirilmesine olanak tanır. Bu diller sayesinde, yazılım geliştirme sürecine katılanlar, mobil uygulama özelliklerini hızlıca test edebilir ve kullanıcı deneyimi üzerinde değişiklikler yapabilir.

Dijital Dönüşüm ve Otomasyon: Çevik yazılım geliştirme, dijital dönüşüm ve süreç otomasyonu, 4GL'ler sayesinde daha hızlı hale gelir. Bu diller, yazılım geliştirme sürecinde organizasyonların dijital dönüşüme daha hızlı uyum sağlamalarına yardımcı olur. Otomasyonun sağlanması, insan müdahalesini minimuma indirerek yazılım geliştirme sürecinde etkinliği artırır.

Daha Az Kod, Daha Yüksek Verimlilik: 4GL'lerin sunduğu en büyük avantajlardan biri, daha az kodla daha hızlı yazılım geliştirme olanağıdır. Geliştiriciler, hazır şablonlar, sihirbazlar ve veritabanı yönetimi araçları kullanarak yazılım uygulamalarını kısa süre içinde oluşturabilirler. Bu sayede yazılım geliştirme süresi önemli ölçüde kısalır ve yazılımın pazara sunulma süresi hızlanır.

Esneklik ve Uyarlanabilirlik: 4GL'ler, geliştiricilerin daha esnek ve uyarlanabilir yazılımlar geliştirmelerini sağlar. Yazılım geliştirme sürecinin her aşamasında yapılabilecek değişikliklere hızlıca yanıt verebilir ve geliştirmeyi optimize edebilir. Bu esneklik, özellikle çevik yazılım geliştirme süreçlerinde önemlidir.

Yeni Nesil 4GL ve Modern Yazılım Geliştirme Yaklaşımları

İnsan-Makine Etkileşimi (HCI) ve 4GL'ler: İnsan ve makine arasındaki etkileşimi geliştiren 4GL'ler, yazılım kullanıcılarının uygulamalarla etkileşimde bulunmalarını daha kolay ve verimli hale getirir. Kullanıcı dostu arayüzler ve basit komutlarla yazılım geliştirme, yeni nesil kullanıcı arayüzlerini kolayca tasarlama imkânı sunar. Bu, yazılımın kullanıcı kabulünü ve başarısını artırır.

Entegre Geliştirme Çerçeveleri ve 4GL'ler: 4GL'ler, günümüzde popüler olan çeşitli yazılım geliştirme çerçeveleri ve araçları ile entegre çalışabilir. Bu, 4GL dilini kullanan yazılımların çok çeşitli sistemlerle uyum içinde çalışmasına imkân tanır. Özellikle DevOps (Geliştirme ve Operasyonlar) süreçlerinde, sürekli entegrasyon ve sürekli dağıtım (CI/CD) için 4GL'ler kullanılabilir.

Veri Görselleştirme ve İleri Düzey Raporlama: 4GL'ler, veri görselleştirme ve raporlama alanlarında da çok etkili araçlar sunar. Bu diller, kullanıcıların verileri hızlıca görselleştirmesine ve anlamlı analizler yapmasına olanak tanır. Çeşitli grafik ve raporlama araçları, büyük veri analizi ile bağlantılı olarak geliştirilebilir.

2.1.5. Altyapı Geliştirme/Edinim Süreçleri

Bilgi sistemleri altyapısı, işletmenin belirli bilgi sistemi uygulamaları için ortam sağlayan teknolojik kaynaklardan oluşur. Donanım, yazılım, danışmanlık, eğitim ve uygulama gibi işletmenin tümü tarafından paylaşılan hizmetlere yönelik yatırımları içerir. Altyapı geliştirme ve edinim süreçleri aşağıdaki tabloda da yer verildiği üzere bünyesinde birçok süreç barındırır. Bu süreçler aşağıda ele alınacaktır.

Altyapı Geliştirme/Edinim Süreçleri Şeması

Fiziksel Mimari Analizinin Proje Aşamaları:

Fiziksel mimari analizi kapsamında gereksinimler, bir kavram kanıtı kullanılarak doğrulanır. Kavram kanıtı, temel kurulum ve işlevselliği gösteren çalışan bir prototip sağlamalıdır. Fiziksel mimari analizine yönelik proje aşamaları ve tedarikçi seçim süreci fiziksel mimari analizinin bir parçasıdır.

a. Mevcut Mimarinin Gözden Geçirilmesi

Mevcut mimarinin gözden geçirilmesi aşamasında, en güncel dokümanların gözden geçirilmesi yapılır. Bir toplantı düzenlenerek, fiziksel mimarideki doğrudan etkilenen tüm alanlardaki uzmanların (genel BT altyapısı, sunucu, depolama, güvenlik vb.) katılımı sağlanmalıdır. Bu aşamada fiziksel mimariyi etkileyen tüm operasyonel kısımların karakterize edilmesine özel dikkat gösterilmelidir. Fiziksel mimarinin aşağıdaki unsurlarına özel dikkat gösterilir:

- Zemin Sorunları: Zemin, altyapının fiziksel temelini oluşturur. Zeminin dayanıklılığı ve istikrarı, sunucular, depolama sistemleri ve diğer ekipmanların güvenli bir şekilde yerleştirilmesi için kritik öneme sahiptir. Toprak koşulları, zemin yapısı ve taşıma kapasitesi gibi faktörler zemin sorunlarının değerlendirilmesinde dikkate alınır.

- Boyut Sınırları: Mevcut altyapının fiziksel boyutları gözden geçirilir. Bu, sunucu rafları, veri depolama sistemleri ve diğer donanım bileşenlerinin ne kadar yer kapladığını ve bu boyutların mevcut tesislerle uyumlu olup olmadığını belirlemek için önemlidir.

- Ağırlık Sınırları: Altyapıdaki cihazlar, sunucular ve depolama sistemleri gibi ekipmanlar belirli ağırlık sınırlarını aşmamalıdır. Bu sınırlar, zeminin taşıma kapasitesini aşmadığından emin olmak için dikkate alınır.

- Akım Güç Kaynağı: Sunucular ve ağ ekipmanları gibi cihazlar için güç kaynakları gözden geçirilir. Bu, enerji gereksinimlerinin karşılanmasını ve kesintisiz bir güç kaynağı sağlanmasını içerir.

- Çevresel Çalışma Sınırları: Sunucular ve veri depolama sistemleri gibi ekipmanlar belirli sıcaklık ve nem koşullarında sorunsuz bir şekilde çalışmalıdır. Bu nedenle, minimum ve maksimum sıcaklık ve nem seviyeleri göz önünde bulundurulur.

- Fiziksel Güvenlik Sorunları: Altyapıdaki cihazlara fiziksel erişimin sınırlanması ve izlenmesi gereken güvenlik gereksinimleri gözden geçirilir. Bu, yetkisiz erişimi önlemeyi ve ekipmanın fiziksel olarak korunmasını içerir.

Burada ilk toplantının çıktısı mevcut altyapı bileşenlerinin ve hedef fiziksel mimariyi tanımlayan kısıtlamaların bir listesidir.

b. Analiz ve Tasarım

Mevcut mimariyi inceledikten sonra, iyi uygulamalara bağlı kalarak ve iş gereksinimlerini karşılayarak, gerçek fiziksel mimarinin analizi ve tasarımı yapılmalıdır.

Bu aşamada aşağıdaki adımlar atılabilir:

- Gereksinimlerin İncelenmesi: İş gereksinimleri ve kullanıcı ihtiyaçları göz önünde bulundurularak, fiziksel mimariye yönelik gereksinimlerin ve beklentilerin belirlenmesi. Bu aşamada işletmenin büyüme planları ve gelecekteki ihtiyaçları da göz önüne alınmalıdır.

- İş Sürekliliği ve Felaket Kurtarma Planları: İş sürekliliği ve felaket kurtarma planlarının tasarımı ve uygulanması. Bu planlar, fiziksel altyapının güvenliğini ve iş sürekliliğini sağlamak için önemlidir.

- Yenilikçi Teknolojilerin Değerlendirilmesi: Mevcut teknolojilerin yanı sıra yenilikçi teknolojilerin analizi ve kullanılabilirliği. Bu, altyapının geleceğe yönelik hazırlıklı olmasını sağlayabilir.

- Altyapı Tasarımı: Fiziksel mimariye uygun bir tasarımın oluşturulması. Bu tasarım, sunucu konumlandırması, veri merkezi düzenlemesi, enerji yönetimi ve soğutma sistemleri gibi unsurları içerir.

- Maliyet ve Bütçe Planlaması: Tasarlanan fiziksel mimari değişikliklerin maliyetinin hesaplanması ve bütçe planlaması yapılması. Bu, projenin finansal yönetimi için kritik önem taşır.

- Güvenlik ve İzleme Stratejileri: Fiziksel güvenlik önlemleri ve sistem izleme stratejilerinin tasarımı. Veri merkezi güvenliği, izin verilen erişim kontrolü ve olay izleme gibi güvenlik unsurları göz önüne alınır.

c. Taslak Fonksiyonel Gereksinimler

Mevcut mimariyi inceledikten sonra iyi uygulamalara bağlı kalarak iş ve gereksinimlerini karşılayarak, gerçek fiziksel mimarinin analizi ve tasarımı yapılmalıdır. Diğer taraftan, taslak fonksiyonel gereksinimler yazılırken, tedarikçi seçim süreci paralel olarak ilerler.

Bu aşamada aşağıdaki adımlar atılabilir:

- Kullanıcı Gereksinimlerinin Belirlenmesi: İşletmenin ve kullanıcıların fiziksel altyapıdan ne beklediğinin belirlenmesi. Bu gereksinimler, performans, güvenlik, kullanılabilirlik ve ölçeklenebilirlik gibi faktörleri içerebilir.

- Fonksiyonel Gereksinimlerin Tanımlanması: Fiziksel altyapının hangi işlevselliği sağlaması gerektiğinin belirlenmesi. Bu, sunucuların işlem gücü, depolama kapasitesi, yedekleme süreçleri, ağ bağlantıları ve diğer temel işlevlerin tanımlanmasını içerir.

- Performans ve Kapasite Gereksinimlerinin Belirlenmesi: Fiziksel altyapının performansının ne olması gerektiğinin ve gelecekteki büyüme için hangi kapasiteye ihtiyaç duyulduğunun belirlenmesi. Bu, iş yükü tahminleri ve veri hacmi artışları gibi faktörleri içerir.

- Güvenlik ve Uyumluluk Gereksinimlerinin Tanımlanması: Fiziksel altyapının güvenlik önlemlerini ve uyumluluk gereksinimlerini karşılaması gerektiğinin belirlenmesi. Bu, veri koruma, erişim kontrolü, yedekleme ve felaket kurtarma planları gibi güvenlik unsurlarını içerir.

- Kullanılabilirlik ve Yedeklilik Gereksinimlerinin Belirlenmesi: Fiziksel altyapının kesintisiz çalışabilirlik ve hızlı kurtarma yeteneklerini içeren kullanılabilirlik ve yedeklilik gereksinimlerinin tanımlanmasını içerir.

- Tedarikçi Seçim Süreci: Aynı zamanda tedarikçi seçim süreci paralel olarak ilerler. Fiziksel altyapının gereksinimlerini karşılayacak uygun donanım ve yazılım tedarikçilerinin belirlenmesi ve değerlendirilmesini kapsar.

d. Fonksiyonel Gereksinimlerin Yazılması

Taslak fonksiyonel gereksinimleri bitirdikten ve bu projenin ikinci kısmını besledikten sonra fonksiyonel gereksinimler belgesi yazılır ve eklenen tüm tarafların personeli ile ikinci mimari toplantısı düzenlenir. Toplantıda bu belge değerlendirilir. Bunun sonucunda tartışılacak ve rafine edilmesi veya eklenmesi gereken gereksinimlerin bir listesi oluşturulacaktır.

Kavram kanıtının (Proof of Concept, POC) planlaması ikinci toplantıdan sonra tamamlanmış fonksiyonel gereksinimlerle başlar.

Bu aşamanın bazı anahtar noktalarına aşağıda yer verilmektedir:

- Fonksiyonel Gereksinimlerin Ayrıntıları: Taslak gereksinimler, daha ayrıntılı ve spesifik hale getirilir. Örneğin, sunucu sayıları, işlemci hızları, bellek kapasiteleri ve ağ bağlantı hızları gibi teknik ayrıntılar dahil edilir.

- Gereksinimlerin Değerlendirilmesi: Fonksiyonel gereksinimler belgesi, projenin tüm tarafları, yani işletme temsilcileri, IT uzmanları ve tedarikçi temsilcileri gibi ilgili personel tarafından gözden geçirilir. Bu toplantıda, belge değerlendirilir ve gereksinimlerin tam, açık ve gerçekçi olduğundan emin olunur.

- Ek Gereksinimlerin Belirlenmesi: İkinci toplantıda, belgede eksik veya daha fazla ayrıntı gerektiren noktalar belirlenir. Bu, gereksinimlerin rafine edilmesi veya eklenmesi gereken yerleri tanımlayan bir liste oluşturulmasını içerir.

- Kavram Kanıtı (Proof of Concept - POC) Planlaması: Fonksiyonel gereksinimler belgesinin hazırlanmasının ardından, projenin karmaşıklığını ve uygulanabilirliğini değerlendirmek için bir kavram kanıtı (POC) planı oluşturulur. POC, projenin belirli bir özelliğini veya bileşenini test etmek ve konseptin geçerliliğini kanıtlamak için kullanılır.

Bu aşama, projenin teknik gereksinimlerinin daha derinlemesine anlaşılmasını sağlar ve projenin başarılı bir şekilde yürütülmesine rehberlik eder. Fonksiyonel gereksinimler belgesi, projenin ilerleyen aşamalarında fiziksel altyapının tasarımı, uygulanması ve test edilmesi için temel bir belgedir.

e. Kavram Kanıtlama (Proof of Concept-POC)

Kavram kanıtlama (POC), önerilen bir ürünün, yöntemin veya fikrin fizibilitesini, gerçek ortamda çalışabileceğini gösterir. Seçilen donanım, yazılım ve güvenlik gereksinimleri dahil tüm beklentileri karşılayabildiğini kanıtlayabilmek için bir POC oluşturulması özellikle tavsiye edilir. POC'nin çıktıları, testleri ve sonuçlarını tanımlayan ilişkili belge ve test protokolleri dahil olmak üzere çalışan bir prototip olmalıdır.

Başlangıçta, POC, aşağıda açıklanan satın alma aşamasının sonuçlarına dayanmalıdır. Bu amaçla, hedef donanımın temsili bir alt kümesi kullanılır. POC'yi çalıştırmak için kullanılan yazılım, test sürümleri veya önceden sağlanan yazılım olabilir. Bu nedenle, ek maliyetlerin minimum olması beklenir. Çerçevenin çoğu öğesi, maliyetleri düşük tutmak için basitleştirilmiş bir biçimde uygulanır. Daha sonraki aşamalarda nihai biçimlerine genişletilecektir.

Kavram kanıtlamanın bazı önemli noktalarına aşağıda yer verilmektedir:

- Donanım ve Yazılım Uyumluluğu: POC, seçilen donanım ve yazılımın projenin ihtiyaçlarına uygun olduğunu doğrulamak için kullanılır. Özellikle, belirli bir işlevi yerine getiren bir prototip oluşturmak, donanımın ve yazılımın projenin gereksinimlerini karşılayabileceğini gösterir.

- Test ve Sonuçlar: POC'nin başarılı bir şekilde çalıştığını ve belirli bir işlevi yerine getirebildiğini gösteren testler ve sonuçlar belgelenir. Bu, projenin geri kalan aşamalarında referans olarak kullanılacak önemli verilere sahip olmayı sağlar.

- Prototip: POC, çalışan bir prototip oluşturulmasını içerir. Bu prototip, projenin ilerleyen aşamalarında daha geniş ve karmaşık hale getirilecek bir temel sunar.

- Maliyet Kontrolü: POC aşamasında ek maliyetler minimumda tutulmaya çalışılır. Bu, projenin başlangıç aşamalarında fazla kaynak harcamadan belirli bir konseptin geçerliliğini doğrulamayı amaçlar.

- Genişletilebilirlik: POC'nin temel amacı, projenin başarılı olabileceğini göstermektir. Bu nedenle, çoğu zaman basitleştirilmiş bir biçimde uygulanır, ancak daha sonraki aşamalarda nihai biçimine genişletilebilir.

Hazırlanan prototip aşağıdaki özellikleri göstermelidir:

- Temel güvenlik altyapısının esas kurulumu,

- Kontrol bileşenlerinin doğru işlevselliği,

- Tanımlanan güvenlik önlemlerinin temel ancak işlevsel uygulaması,

- Güvenli işlemler,

- Kurulum kısıtlamaları ve sınırları (sunucu boyutu, sunucu akım tüketimi, sunucu) açısından karakterizasyon ağırlık, sunucu odası fiziksel güvenliği),

- Performans,

- Operasyonel durumda temel yük devretmeyi dahil etmek için güvenilir esneklik,

- Finansman ve maliyetlendirme modeli,

- Veri ve algoritma.

Dağıtım için hazırlanan ilgili uygulama projeleri de POC'nin bir parçası olmalıdır. Bunun nedeni bunların üretimin fiziksel mimarisinde kullanılanlar ile aynı nitelikte kullanılacak olmalarıdır. Bu aşamanın sonunda, üretimin boyutlandırılması ve düzeni için POC sonuçlarını içerecek şekilde uyarlandığı son bir toplantı yapılır.

İşletme, uygulamaların dağıtımı ve işletimi için bir dış kaynak kullanımı/sınır ötesi modelle uğraşıyorsa, ek hususlar geçerli olabilir. Bununla beraber, BT ortamını çalıştırma platformu da (sahip olunan, bulut tabanlı, sanallaştırma) ek hususlar getirebilir. Örneğin, işletme yüksek düzeyde düzenlemeye tabi bir sektörde veya yüksek kullanılabilirlik gerektiren bir sektörde faaliyet gösteriyorsa, POC test edilirken veri gizliliğini sağlamak için yeterli fazlalık ve güvencenin dikkate alınması gerekebilir.

Altyapının Uygulamaya Konmasının Planlanması:

Altyapı uygulamasının planlanmasında, sonuçların kalitesini sağlamak aşamalı bir yaklaşım kullanılması gerekir. Diğer projelerle iletişim süreçleri oluşturmak da önemlidir. Bu farklı aşamalar sayesinde, bileşenler bir araya getirilir ve tedarik aşamasında ve sonrasında seçim süreci kullanılarak mevcut ve iletişim kurulabilecek tedarikçilerin net bir şekilde anlaşılması sağlanır. Ayrıca, teslimat, kurulum ve test planlarının geliştirilmesini içeren sonraki adımları hazırlamak için temel iş ve teknik gereksinimlerin kapsamını seçmek gerekir. Diğer taraftan, geleceğe yönelik bir çözüm sağlamak için doğru becerilere sahip doğru ortakları seçmek çok önemlidir. İhtiyaç analizi bu sürecin bir parçası değildir. Ancak sonuçları sürekli olarak sürece besler. Bu aşamalarla bir Gantt şeması üretilirse, bazı aşamalar muhtemelen çakışacaktır. Bu nedenle, farklı aşamalar yinelemeli bir süreç olarak düşünülmelidir.

Aşağıdaki dört farklı aşamada sonraki projelere hazırlanmak için tüm bileşenlerin birbirine uyması gerekir (örn. veri geçişi).

1. Tedarik Aşaması

Tedarik aşamasında, seçilen çözüme genel bir bakış sağlamak ve çıktıların nicel yapısını belirlemek için işletme ve analiz projesi arasında iletişim kurulur. Gereksinim ifadeleri de üretilir. Ayrıca tedarik süreci, hizmet düzeyi yönetimi sürecini başlatır. Bu faaliyetler sırasında, tercih edilen ortaklar müzakere sürecine davet edilir ve teslimatlar, sözleşmeler ve SLA'lar[1] imzalanır. Özetle bu aşamanın içeriğini;

- İletişim surecini oluşturma ve teslimatları, sözleşmeleri ve SLA'leri belirleme ve

- Gereksinim beyanını üretme

oluşturur.

2. Teslimat Süresi

Teslimat süresi aşamasında teslimat planı geliştirilir. Bu aşama, bazı kısımlarda satın alma aşamasıyla çakışır. Teslimat planı, öncelikler, hedefler ve gayri resmi hedefler, temel gerçekler, ilkeler, iletişim stratejileri, kilit göstergeler ve kilit görev ve sorumluluklardaki ilerleme gibi konuları içermelidir. Bu aşamanın içeriğini teslimat planını geliştirme kapsamında öncelikler, amaçlar, önemli gerçekler, ilkeler, iletişim stratejileri, anahtar göstergeler, önemli işlerdeki ilerlemeler ve sorumluluklar oluşturur.

3. Kurulum Planı

Kurulum planlama aşamasında, kurulum planı etkilenen tüm taraflarla iş birliği içinde geliştirilir. Ek bir adım, planı ilgili taraflarla ve tabii ki entegrasyon projelerinden sorumlu olanlarla gözden geçirmektir. Bu yinelemeli bir süreçtir. Bu aşama ise özetle ilgili taraflarla planı oluşturma ve gözden geçirme şeklinde tanımlanabilir.

4. Kurulum Test Planı

[1] *SLA, müşterinin Plan(lar)ının sürekliliğinin korunması için müşterinin, VeriTeknik'e yüklemiş olduğu sınırlı sorumlulukların tümüne verilen kısa isimdir.*

Kurulum planının bilinen bağımlılıklarına dayalı olarak test planı geliştirilir. Test planı, mümkün olduğunda uygulamalar ve altyapı için test senaryolarını, temel gereksinimlerin özelliklerini, süreçlerin tanımını ve ölçüm bilgilerini içerir. Projenin ilk kısmı (fiziksel mimarinin analizi) tamamlanmalı ve gerekli altyapı kararları alınmalıdır. Son aşama ise test senaryolarını, temel gereksinim spesifikasyonları, süreçlerin ve ölçütlerin tanımlanmasını içeren test planı geliştirme şeklinde verilebilir.

2.1.6. Donanım/Yazılımı Tedarik Süreci

Donanım ve yazılım tedarik sürecinde aşağıda belirtilen hususlar dikkate alınır.

Donanım Edinimi:

a. Sistem Edinim Etkenleri

Bir sistemi geliştirmek mi yoksa satın almak mı kararı için aşağıdaki etkenler değerlendirilmelidir:

- Sistemin İşlevsel Olması Gerektiği Tarih: İşlevsel sistemin ne zaman gerektiği belirlenmelidir. Acil ihtiyaçlar veya projenin zaman çizelgesi bu faktörü etkiler.

- Sistemi Satın Almak Yerine Sistemi Geliştirme Maliyeti: Sistemi geliştirmek ve özelleştirmek, bir sistem satın almak kadar maliyetli olabilir. Bu maliyetleri dikkate almak önemlidir.

- Gereken Kaynaklar, Personel ve Donanım: Sistemi geliştirmek için yeterli kaynak, personel ve donanımın olup olmadığı değerlendirilmelidir.

- Tedarikçi Sistemindeki Lisanslama Özellikleri: Satın alınacak sistemin lisanslama koşulları, maliyetler ve sürekli bakım gereksinimleri dikkate alınmalıdır.

- Yeni Sistemle Arayüz Oluşturmaya İhtiyaç Duyacak Diğer Sistemler: Yeni sistem, mevcut sistemlerle veya diğer yazılımlarla nasıl etkileşimde bulunacak ve arayüz oluşturma gereksinimleri değerlendirilmelidir.

- Stratejik İş Planları, Risk İştahı ve Yasal Uyumluluk: Sistem tedarik kararı, işletmenin stratejik planları, risk toleransı ve yasal uyumluluk gereksinimleri ile uyumlu olmalıdır.

- İşlevsellik Değişiklikleri için Gelecekteki Olası Gereksinimler: Sistemin ilerleyen dönemlerdeki işlevsellik değişiklikleri için esneklik sağlaması önemlidir. Gelecekteki gereksinimler göz önünde bulundurulmalıdır.

b. Sistem Spesifikasyonları

Yeni bir sistem edinirken, teknik özellikler aşağıdakileri içermelidir:

- Organizasyonel Tanım: Sistemin organizasyon içindeki rolü ve yapısı net bir şekilde tanımlanmalıdır. Merkezi, merkezi olmayan, dağıtılmış, dış kaynak kullanımı gibi organizasyonel özellikler bu tanımda yer almalıdır.

- Güvenlik Sağlamlığı İçin Donanım ve Yazılım Değerlendirme Güvencesi Seviyeleri: Sistemin güvenlik gereksinimleri ve bu gereksinimleri sağlamak için kullanılacak donanım ve yazılımın değerlendirme güvencesi seviyeleri açıkça belirtilmelidir.

- Bilgi İşleme Gereksinimleri: Sistem tarafından işlenecek bilgilerin türü, miktarı, işleme süreçleri ve gereksinimleri ayrıntılı olarak açıklanmalıdır.

- Donanım Gereksinimleri: Sistem tarafından kullanılacak donanımın türü, kapasitesi, performans özellikleri ve bağlantı gereksinimleri belirtilmelidir.

- Sistem Yazılım Uygulamaları: Sistem tarafından kullanılacak yazılım uygulamaları, bunların özellikleri, versiyonları ve uyum gereksinimleri açıkça ifade edilmelidir.

- Destek Gereksinimleri: Sistemin işletilmesi, bakımı ve desteklenmesi için gerekli kaynaklar, personel, yazılım güncellemeleri ve yedekleme gereksinimleri belirtilmelidir.

- Uyarlanabilirlik ve Dönüşüm Gereksinimleri: Sistemin gelecekteki ihtiyaçlara nasıl uyum sağlayacağı ve gerektiğinde nasıl dönüştürülebileceği açıkça tanımlanmalıdır.

- Sistem Kısıtlamaları: Sistem tarafından karşılanması beklenmeyen veya mümkün olmayan kısıtlamalar, bu aşamada net bir şekilde belirtilmelidir.

Bu alanda bir denetim yaparken, bilgi sistemleri denetçisi aşağıdakileri de dikkate almalıdır:

- Satın alma işleminin bir iş gereksinimi ile başlayıp başlamadığını ve bu ihtiyaç için donanım gereksinimlerinin spesifikasyonlarda dikkate alınıp alınmadığını belirlemek.

- Birkaç işletmenin değerlendirilip değerlendirilmediğini ve aralarındaki karşılaştırmanın belirtilen kriterlere göre yapılıp yapılmadığını belirlemek.

Yazılım Edinimi:

a. Gereksinimlerin Tanımı

Gereksinimlerin tanımı, bir sistemin ne yapması gerektiği, kullanıcıların bir sistemle nasıl etkileşime gireceği, sistemin hangi koşullarda isleyeceği ve sistemin yerine getirmesi gereken bilgi kriterleri hakkında açıklamalar içermelidir. Gereksinimler tanımlanırken ihtiyaç analizi çıktıları, performans gereksinimleri, tasarım gereksinimleri ve müşteri gereksinimleri de göz önünde bulundurulmalıdır.

Yazılım edinimi aşamasında, sistemin hangi yazılımın kullanılacağına ve bu yazılımın hangi gereksinimleri karşılayacağına dair ayrıntılı bir tanım yapılmalıdır. Bu tanım, projenin başarılı bir şekilde ilerlemesi için büyük bir öneme sahiptir. Gereksinimlerin tanımı aşağıdaki açıklamaları içermelidir:

- Sistemin Fonksiyonları: Yazılımın ne yapması gerektiği, kullanıcıların hangi işlevleri gerçekleştireceği ve bu işlevlerin nasıl yerine getirileceği açıklanmalıdır.

- Kullanıcı Etkileşimi: Kullanıcıların sistemi nasıl kullanacakları, kullanıcı arabirimi tasarımı ve kullanıcı deneyimi gereksinimleri açıkça tanımlanmalıdır.

- Çalışma Koşulları: Yazılımın hangi koşullarda kullanılacağı, çalışma ortamı gereksinimleri ve performans beklentileri belirtilmelidir.

- Bilgi Kriterleri: Yazılımın işleyişine ve sonuçlarına dair belirli bilgi kriterleri açıklanmalıdır. Bu, veri girişi, çıkışı ve işleme ile ilgili olabilir.

Gereksinim tanımlarını başarıyla tamamlamak için proje ekibi aşağıdaki gibi görevleri yerine getirmelidir:

- Mevcut Kısıtlamaları Belirleme: Projeye etki edebilecek mevcut kısıtlamaların net bir şekilde belirlenmesi.

- Paydaşların Belirlenmesi: Projeye dahil olan veya etkileyen tüm paydaşların belirlenmesi ve onların gereksinimlerinin anlaşılması.

- Uyuşmazlık ve Uygunsuzlukların Belirlenmesi ve Çözülmesi: Gereksinimler arasındaki uyuşmazlıklar veya uyumsuzluklar tespit edildiğinde, bu sorunların çözülmesi.

- Gereksinimlerin Yapısal Şekilde Belirlenmesi: Gereksinimlerin ayrıntılı ve yapısal bir şekilde belirlenmesi ve bu gereksinimlerin tüm paydaşların onayına sunulması.

- Uyuşmazlık ve Aksaklıkların Çözülmesi: Gereksinimler arasındaki uyuşmazlıkların veya aksaklıkların çözülmesi ve gereksinimlerin daha tutarlı hale getirilmesi.

- Gereksinimlerin Doğrulama: Gereksinimlerin eksiksiz, tutarlı, açık, doğrulanabilir, değiştirilebilir, test edilebilir ve izlenebilir olduğunun doğrulanması.

b. Teklif Talebi (RFP)

Teklif talebi (RFP), satıcılardan müşterinin sorunlarına veya iş gereksinimlerine çözüm önermelerini isteyen bir belgedir. Bunlardan bazıları iş hakkında, yasal ve güvenlikle ilgili

gereksinimlerin yanı sıra geliştirme ile ilgili gereksinimlerdir. Talep, tanımlanmış bir ürün veya hizmet için farklı tedarikçilerden ayrıntılı ve karşılaştırılabilir teklifler almak için resmi bir yöntemdir. Satın alma kararına ilişkin gerekli tüm bilgileri içeren kapsamlı ve resmi belgedir.

RFP süreci boyunca tedarikçilere karşı şeffaf bir tutumla ihtiyacın net bir şekilde dile getirilmiş olması kritik öneme sahiptir. Gizlilik kaygısı duyulan bir proje ise, masaya oturması muhtemel tüm tedarikçilerle gizlilik sözleşmesi imzalanabilir.

Gereksinimlerin net olarak tanımlanması için bir teklif talebi hazırlanır. Teklif talebi hazırlanırken aşağıdaki unsurların değerlendirilmesi zorunludur.

Teklif Talebi Unsurları Şeması

Yazılım edinmeye ilişkin olarak bilgi sistemleri denetçisi aşağıdakileri yapmalıdır:

- Sözleşme İncelemesi: İmza atılmadan önce, yazılım tedarik sözleşmesinin hukuk danışmanı tarafından kontrol edildiğinden emin olunmalıdır. Bu, sözleşmedeki taahhütlerin ve şartların yasal olarak geçerli olduğunu ve kurumun haklarını koruduğunu doğrulamak için önemlidir.

- RFP İncelemesi: RFP belgesi, projenin gereksinimlerini ve beklentilerini belirler. Bilgi sistemleri denetçisi, seçilen tedarikçinin sunacağı çözümün RFP belgeleri tarafından desteklenip desteklenmediğini kontrol etmelidir.

- Fizibilite Çalışması İncelemesi: Yazılımın edinilmesi kararı, önceden bir fizibilite çalışması tarafından desteklenmelidir. Bu dökümantasyon, projenin maliyet etkinliği, uygunluk ve iş gereksinimlerini karşılayıp karşılamadığını değerlendirmek için önemlidir.

- Sunum ve Pilot Uygulama Katılımı: Seçilen tedarikçi, projenin başarılı bir şekilde uygulanması için gündem temelli sunumlar ve konferans salonu pilot uygulamaları yapabilir. Bilgi sistemleri denetçisi, bu etkinliklere katılarak sistemin iş gereksinimlerine uygun olduğunu doğrulamalıdır.

- Sözleşme Gözden Geçirme: Sözleşme imzalanmadan önce, tedarikçi ile yapılan sözleşme metni dikkatlice gözden geçirilmelidir. Bu, tarafların taahhütlerini ve sorumluluklarını belirler.

- Güvenlik İncelemesi: Bilgi sistemleri denetçisi, yazılım tedarik sürecinde güvenlik müdahale ve sorumluluklarının sözleşmeye dahil edilip edilmediğini kontrol etmelidir. Güvenlik, herhangi bir yazılım sistemi için kritik bir faktördür.

- Fizibilite Çalışması Analizi: Yazılım edinme kararının uygunluğunu belirlemek için fizibilite çalışmasından gelen dokümantasyon analiz edilmelidir. Bu, projenin maliyet, fayda, risk ve uygunluk açısından değerlendirilmesini içerir.

- RFP Uyumluluğu: Seçilen tedarikçinin RFP belgeleriyle uyumlu bir teklif sunup sunmadığını doğrulamak önemlidir. Bu, belirtilen iş gereksinimlerinin karşılandığını gösterir.

- Ürün Kapsamı Kontrolü: Sözleşme özelliklerinin RFP belgelerine uygun olduğundan ve belirtilen ürünleri içerdiğinden emin olunmalıdır.

Bilgi sistemleri denetçisi, bu görevleri yerine getirerek yazılım edinme sürecinin sağlam bir şekilde yürütülmesine katkıda bulunur ve kurumun projenin başarılı bir şekilde sonuçlanmasını sağlamada önemli bir rol oynar.

Teklif talebi (RFP) süreci, yazılım edinimi sürecinin çok kritik bir aşamasıdır ve bu aşamada dikkat edilmesi gereken noktalar, sürecin başarılı bir şekilde tamamlanmasında önemli bir rol oynar. Aşağıda RFP süreci ile ilgili olarak dikkat edilmesi gereken önemli hususlar yer almaktadır:

- Yazılım Tedarik Süreci İyileştirme: RFP süreci yalnızca yazılım tedarik sürecinin başından sonuna kadar doğru bir şekilde yönetilmesi gereken bir aşama değil, aynı zamanda yazılım seçiminde optimizasyon yaparak en verimli sonucu almanızı sağlar. Bu süreçte, yazılım sağlayıcılarının sağlayabileceği sürdürülebilir destek ve uzun vadeli ilişkiler değerlendirilmeli ve yazılım tedarikçisinin geçmiş performansı, desteği ve iş sürekliliği sağlama yeteneği göz önünde bulundurulmalıdır.

- Yapay Zeka Destekli Seçim Süreçleri: Son yıllarda yapay zeka ve makine öğrenimi tabanlı araçlar, yazılım seçim süreçlerini daha verimli ve veri odaklı hale getirmek için kullanılmaktadır. Özellikle büyük ölçekli yazılım edinim projelerinde, bu tür teknolojiler kullanılarak daha isabetli seçimler yapılabilir. Yapay zeka destekli analizler, çok sayıda yazılım çözümünü hızlı bir şekilde karşılaştırmak ve yazılımın performansını simüle etmek için kullanılabilir.

- Bulut Tabanlı Yazılımlar için RFP Hazırlığı: Bulut bilişim, modern yazılım tedarik süreçlerinde önemli bir yer tutmaktadır. Bulut tabanlı yazılımlar için teklif talepleri hazırlarken, yazılımın bulut ortamlarında çalışma kabiliyeti, veri güvenliği önlemleri ve sağlayıcının bulut servislerinin kesintisizliğini sağlama kapasitesi de dikkate alınmalıdır. Bu, yazılım edinimi sürecine yeni bir boyut ekleyerek yazılımın iş sürekliliğine olan katkısını artırır.

- Bilişim Güvenliği Değerlendirmeleri: Yazılım tedarikinde özellikle güvenlik, kritik bir faktördür. Bu bağlamda, RFP hazırlığında siber güvenlik gereksinimlerinin net bir şekilde belirtilmesi gerekmektedir. Tedarikçilerin sağladığı güvenlik önlemleri, veri şifreleme, erişim kontrol sistemleri ve düzenli güvenlik denetimleri gibi unsurların titizlikle değerlendirilmesi ve sözleşme metnine yansıtılması gerekir.

- RFP İle İlgili Eğitim ve Geliştirme: Yazılım edinimi sürecinde yer alan tüm ekip üyelerinin ve proje yöneticilerinin, RFP süreci hakkında eğitim almaları önemlidir. Bu eğitimler, tedarikçi seçiminden sözleşme imzalamaya kadar her aşamada şeffaflık, etkililik ve verimlilik sağlamak için gereklidir. Ayrıca, RFP'nin uygun şekilde hazırlanmaması durumunda proje yönetiminde yaşanabilecek aksaklıklar da göz önünde bulundurulmalıdır.

- Veri Gizliliği ve Yasal Uygunluk: Özellikle düzenlemelere tabi olan yazılım edinim projelerinde, veri gizliliği ve yasal uyumluluk çok büyük önem taşır. RFP hazırlarken, yazılımın yerel ve uluslararası veri koruma yasalarına uygunluğu, veri işleme süreçleri ve kullanıcı verilerinin korunması gibi faktörlerin açıkça belirtilmesi gereklidir.

2.1.7. Test Süreçleri

Yazılım ve donanım edimini sırasında, söz konusu edinimlerin testlerinde çeşitli yöntemler kullanılır. Bu yöntemlere aşağıda yer verilmektedir.

Test Kategorileri:

a. Birim Test

Birim testi, bir başka yazılımın işlevlerini (metot, sınıf vs.) çalıştıran yazılım yöntemidir. Yazılım, birim testi ile test edilen kodun beklenen duruma (durum testi) gelip gelmediğini veya olayların oluş sırasının (davranış testi) çalışıp çalışmadığı kontrol edilir.

Belli bir program veya modülün program mantığını test eder. Programın iç operasyonunun spesifikasyonları göre işlemesini sağlar. Prosedürel tasarımın kontrol yapısına odaklı bir dizi test senaryosu kullanır.

Birim test yönteminin avantajlarına aşağıda yer verilmektedir:

- Hataların Erken Tespit Edilmesi: Yazılımın belirli birimlerini doğru bir şekilde test ederek hataları erken aşamalarda yakalamanıza yardımcı olur. Bu, hataların daha sonra daha pahalı ve zaman alıcı aşamalarda düzeltilmesini engeller.

- İzolasyonun Sağlanması: Birimi izole eder ve diğer bileşenlerden bağımsız olarak çalıştırır. Bu, birimlerin ayrı ayrı test edilebilmesini ve hataların daha kolay tanımlanabilmesini sağlar.

- Otomasyonun Hızlandırılması: Genellikle otomatik olarak çalıştırılır, bu da test sürecini hızlandırır ve geliştiricilerin manuel olarak her testi çalıştırma zorunluluğunu ortadan kaldırır.

- Hızlı Geri Bildirim: Birim testi hızlıdır ve hızlı geri bildirim sağlar. Bu, geliştiricilerin kod değişikliklerinin sonuçlarını hemen görmelerini ve hataları hızlıca düzeltmelerini sağlar.

- Yazılım Kalitesinin Artırılması: Yazılımın kalitesini artırır. Beklenen davranışları tanımladığı için yazılımın daha güvenilir ve istikrarlı olmasını sağlar.

- Spesifikasyonlara Uygunluk: Yazılımın belirli gereksinimlere ve spesifikasyonlara uygun olduğunu doğrular. Bu, yazılımın kullanıcı ihtiyaçlarını karşıladığından emin olmanıza yardımcı olur.

- Hata Ayıklama Kolaylığı: Bir birim testi başarısız olduğunda, hatanın kaynağını daha kolay belirleyebilir. Bu, hataları hızlıca düzeltebilmesini sağlar.

- Dokümantasyon: Test ettiği kodun dokümantasyonunu sağlar. Karışık bir algoritma içeren bir fonksiyonu, kaynak kodundan daha kolay bir şekilde birim testleri okunarak anlaşılabilir.

- Yeniden Kullanılabilirlik Artışı: Yazılımın farklı projelerde veya farklı bağlamlarda yeniden kullanılabilir olmasına yardımcı olur. Bu, geliştiricilerin daha önce yazdıkları kodları başka projelerde tekrar kullanmalarına olanak tanır.

Birim testleri, yazılım geliştirme sürecinin önemli bir parçasıdır ve yazılımın daha güvenilir, kaliteli ve sürdürülebilir olmasına katkıda bulunur.

Birim testleri, genellikle birim test çerçeveleri (örneğin, JUnit, NUnit, pytest gibi) kullanılarak uygulanır ve yazılım geliştirme sürecinin önemli bir parçasını oluşturur. Bu testler, yazılımın güvenilir ve istikrarlı bir şekilde çalışmasını sağlamak için çok önemlidir.

Birim test yönteminin dezavantajlarına aşağıda yer verilmektedir:

- Zaman ve Kaynak İhtiyacı: Birim testleri yazmak, bakım yapmak ve çalıştırmak zaman ve kaynak gerektirir. Büyük projelerde veya sık sık değişen kod tabanlarında, birim testleri sürdürmek zor olabilir.

- Kod Duplikasyonu: Birim testleri genellikle kodun kendisiyle benzerdir ve bazı kod duplikasyonuna neden olabilir. Bu, bakım maliyetini artırabilir çünkü birim testleri de güncellenmelidir.

- Tam Kapsama Sağlama Zorluğu: Tüm olası kod yollarını kapsayan eksiksiz bir birim test seti oluşturmak zor olabilir. Bazı durumlarda, tüm kodun tam kapsamasını sağlamak mümkün olmayabilir.

- Kötü Birim Test Tasarımı: Kötü birim test tasarımı, birim testlerin karmaşıklığını artırabilir ve bakımı zorlaştırabilir. Doğru test senaryolarını tasarlamak önemlidir.

- Değişen Kodlara Uyum Sağlama: Yazılımın gereksinimleri veya kod tabanı değiştikçe, birim testlerini güncellemek veya düzeltmek gerekebilir. Bu, zaman alıcı olabilir.

- Yavaş Geliştirme Süreci: Birim testleri yazmak ve çalıştırmak, geliştirme sürecini yavaşlatabilir. Özellikle testler otomatik olarak çalıştırılmıyorsa, süreç daha da uzun sürebilir.

- İlk Başta Ek Çaba Gerektirir: Yazılım geliştirme başlangıcında birim testleri yazmak ve otomatize etmek ek bir çaba gerektirir. Bu, projenin başlangıcında ek zaman alabilir.

- Yanlış Pozitifler veya Yanlış Negatifler: Birim testleri yanlış pozitif sonuçlar (hatalar bildirirken aslında hata olmayan durumlar) veya yanlış negatif sonuçlar (gerçek hataları kaçırır) verebilir. Bu, güvenilir birim testleri oluşturmanın önemini vurgular.

- Bağımlılık Sorunları: Birim testleri, dış kaynaklara (veritabanları, web servisleri, harici API'lar) bağımlıysa, bu bağımlılıkların yönetilmesi ve test edilmesi karmaşık olabilir.

Birim testleri, yazılım geliştirme sürecinin önemli bir parçasıdır ve birçok avantaj sunar, ancak bu dezavantajları da göz önünde bulundurarak dikkatlice planlanmalı ve uygulanmalıdır. İyi bir birim test stratejisi, bu dezavantajları minimize etmeye yardımcı olabilir.

b. Arayüz ve Entegrasyon Testi

Arayüz testi, bir alandan diğerine bilgi ileten iki veya daha fazla bileşenin bağlantısını değerlendiren bir donanım veya yazılım testidir. Entegrasyon testi veya genel test olarak da bilinen bu test yöntemi birden fazla modül veya bileşeni olan sistemin tümünü kontrol eden donanım veya yazılım testidir. Genelde birim testleri (unit test) biten yazılımlar için uygulanır. . Entegrasyon testi, yazılım geliştirme sürecinin ileri aşamalarında uygulanır. Çünkü birim testleri (unit test) tamamlandığında bileşenlerin bir araya getirilmesi gerekir. Entegrasyon testi bir şeyin çalışıp çalışmadığını söylerken, birim testi neden çalışmadığını söyler. Birim testi yazılımcı perspektifinden bakarken, entegrasyon testi kullanıcı perspektifinden yazılır.

Entegrasyon testindeki asıl amaç, oluşturulan yazılım modüllerinin bir araya getirilerek doğruluğunun sağlanmasıdır. Yazılımın bileşenlerinin ve modüllerinin birleştiğinde beklenen işlevselliği ve uyumu sürdürebildiğinden emin olmaktır. Yazılım ürünü için oluşturulan tüm modüller bir araya getirilir ve bu şekilde test edilir. Burada ki amaç, metotlar birim başına testten geçerken, modüller halinde bir araya geldiğinde bazı hatalara sebep oluyor olabilirler. Entegrasyon testleri, bu tarz yazılım ürünü problemlerinin henüz canlı (prod) ortama çıkmadan tespit edilmesini veya geliştirilen yeni modülün de sorunsuz çalışabileceğinden hızlı bir şekilde emin olunmasını sağlar.

Entegrasyon testi için farklı yöntemler kullanılabilir. Bunlar;

Büyük Patlama Entegrasyon Testi (Big Bang Integration Test): En yaygın kullanılan entegrasyon test tipidir. Tüm bileşenler bir araya getirilerek tüm sistem test edilir. Hızlı ve kolay bir şekilde beraber çalıştıklarında anlam ifade eden modüllerin doğruluğunu sağlar. Ancak birim başı metot doğruluğunun gözden kaçınılması mümkündür. Bileşenler arasındaki etkileşimleri test etmek zor olduğu için, hata tespiti ve düzeltme süreci uzun olabilir.

Büyük Patlama Entegrasyon Testi, tüm sistemin bir araya getirilmesiyle yapılan bir test türüdür. Bu testte, tüm bileşenler birleştirilip tek bir aşamada test edilir. Bu yöntem genellikle hızlı ve kolay bir şekilde tüm sistemin uyumlu çalışıp çalışmadığını görmek amacıyla kullanılır. Ancak, bu testin bazı dezavantajları da bulunmaktadır:

- Hızlı Uygulama: Büyük Patlama testinin en belirgin avantajı, tüm sistemin hızlıca test edilmesidir. Tüm bileşenlerin bir arada çalışıp çalışmadığını görmek, testin ilk aşamasında çabuk bir şekilde elde edilebilir. Bu tür testler genellikle tüm bileşenlerin birbirleriyle entegrasyonu sağlandığında, hızlı bir şekilde modüllerin doğruluğunu kontrol etme imkânı sunar.

- Modüller Arası Etkileşimler: Ancak, bu test türü bileşenler arasındaki etkileşimleri test etmekte zorlanabilir. Bileşenler bir araya getirildiğinde, her bir bileşenin bağımsız olarak doğruluğu test edilmediği için etkileşimde meydana gelebilecek hataların kaynağını bulmak zorlaşır. Bu nedenle, hata tespitinde önemli zorluklar yaşanabilir.

- Uzun Hata Tespiti Süreci: Büyük Patlama testinin bir başka dezavantajı, hata tespitinin uzun sürebilmesidir. Çünkü tüm bileşenler bir arada çalışmaya başladığında, modüller arasındaki etkileşimlerde yaşanacak hataların nereden kaynaklandığını belirlemek, birim başı doğruluk kontrolü yapılmadığı için zorlaşır. Bu, hata ayıklama sürecini uzatır ve daha fazla kaynak gerektirir.

- Düşük İzolasyon: Bu tür testlerde modüllerin her biri bağımsız olarak test edilmediği için, bileşenler arasında yüksek düzeyde bir bağımlılık söz konusu olur. Her modül, diğer modüllerle olan etkileşiminde beklenmedik hatalarla karşılaşabilir. Bu da, sistemin her yönünün doğru bir şekilde çalışıp çalışmadığını tek seferde anlamayı zorlaştırabilir.

Tümdengelim Entegrasyon Testi (Top-Down Integration Test): Sistemdeki ana bileşenlerden başlanarak kademeli olarak diğer bileşenlere doğru test yapılır. Böylece, hataların daha

erken tespit edilmesi ve düzeltilmesi sağlanır. Burada amaç, modüller arası geçiş yapılırken hatalı olan modülün kolay bir şekilde bulunabilmesidir. Test işlemi yukarıdan aşağı doğru olmakta ve her modülün test işleminden başarılı bir şekilde geçerek ilerlemesi gerekmektedir.

Tümdengelim Entegrasyon Testi (Top-Down Integration Test), yazılım geliştirme sürecindeki test aşamalarını daha verimli hale getiren ve yazılımın her bir bileşeninin entegrasyonunu dikkatlice analiz eden önemli bir yöntemdir. Bu test türü, sistemdeki ana bileşenlerden başlanarak kademeli bir şekilde diğer bileşenlere doğru ilerler ve hataların erken tespit edilmesine olanak tanır. Bu yöntemi daha etkili kılmak için dikkat edilmesi gereken bazı önemli hususlar şunlardır:

- Erken Hata Tespiti: Tümdengelim testi, hata tespitini erken aşamalarda gerçekleştirmenize olanak tanır. Sistemin ana bileşenleri test edilip onaylandıktan sonra alt modüllere geçilir, bu da erken hata tespiti sağlar. Bu, yazılımın daha sağlam olmasına yardımcı olur.

- Üst Seviye Modüllerin Testi: Bu test türü, ilk olarak üst seviye modüllerin test edilmesine olanak tanır. Yüksek öncelikli işlevselliklerin doğruluğu kontrol edildikten sonra, daha düşük seviyedeki bileşenler test edilmeye başlanır. Bu, kullanıcıların en kritik işlevselliklerin sağlıklı çalıştığını hızlıca doğrulamasını sağlar.

- Hatalı Modüllerin Kolay Tespiti: Modüller arasındaki geçişlerde hatalı olan modül kolayca bulunabilir. Herhangi bir hata tespit edildiğinde, hata ilk etapta yüksek seviyede olduğu için, ilgili modülün altındaki bileşenlerin test edilmesine gerek kalmadan problemi çözmek mümkündür.

- Test Sürecinin Kolaylaştırılması: Tümdengelim testinin avantajlarından biri de test sürecinin daha düzenli ve yönetilebilir hale gelmesidir. Üst modüller doğrulandıktan sonra alt modüller test edilip, yazılımın her bir parçası sırasıyla denetlendikten sonra, her bileşenin uyum içinde çalıştığı onaylanır.

Tümevarım Entegrasyon Testi (Bottom-Up Integration Test): Sistemdeki alt seviyedeki bileşenlerden başlanarak üst seviyedeki bileşenlere doğru test yapılır. Birim testler ile beraber ilerler. Alt kademedeki modüller birim testlerden geçirilerek yukarıya doğru ilerlenir. Her testin başarılı olması gerekmektedir. Buradaki amaç, en alt modüllerden başlayarak hataların en kısa sürede bulunabilmesidir. Bu şekilde, bileşenler arasındaki etkileşimler daha iyi test edilir.

Tümevarım Entegrasyon Testi (Bottom-Up Integration Test), yazılım geliştirme sürecinde modüllerin alt seviyelerden başlanarak yukarıya doğru test edildiği etkili bir test yöntemidir. Bu test türü, özellikle yazılımın alt seviyedeki bileşenlerinin doğruluğunu sağlamak ve hataların en erken aşamalarda tespit edilmesini kolaylaştırmak için kullanılır. Bu yöntemi daha verimli hale getirmek için aşağıdaki noktalar önemlidir:

- Alt Seviye Modüllerin Hızlı Testi: Tümevarım testinin temel avantajlarından biri, alt seviyedeki modüllerin hızlı bir şekilde test edilmesidir. En alt seviyedeki bileşenler doğrulandıktan sonra, yazılımın daha üst seviyelerine doğru geçilir, bu da hata tespitini hızlandırır.

- Bileşenler Arası Etkileşimin İyi Test Edilmesi: Tümevarım testi, bileşenler arasındaki etkileşimleri daha net bir şekilde test edebilme fırsatı sunar. Alt seviyedeki modüller başarılı bir şekilde doğrulandıktan sonra, üst seviyedeki bileşenlerin uyum içinde çalışması kolaylıkla gözlemlenir.

- Modüllerin Bağımsız Test Edilmesi: Bu test yöntemi, her modülün bağımsız olarak test edilmesini sağlar. Alt seviyedeki bileşenler başarılı bir şekilde doğrulandıktan sonra, üst seviyedeki modüllerle etkileşimlerini test etmek için daha az bağımlılık olur, bu da test sürecini daha verimli hale getirir.

- Hata Ayıklamanın Kolaylaşması: Tümevarım entegrasyon testinde, hatalar erken aşamalarda tespit edilir ve üst seviyedeki modüllerin doğruluğu kontrol edildikten sonra hatalı modüllerin tespiti daha hızlı yapılır. Bu sayede yazılımın hata ayıklama süreci daha kısa sürede tamamlanabilir.

- Test Sürecinin Sistematik Olması: Alt seviyedeki modüller test edildikten sonra, üst seviyedeki bileşenlerle entegrasyon testlerine geçilir. Bu sıralı test yapısı, sistemin her parçasının doğruluğunu tek tek kontrol etmeye olanak tanır ve daha düzenli bir test süreci sağlar.

Karma Entegrasyon Testi (Sandwich/Hybrid Integration Test): Sistemdeki modüllerin bir kısmının tümdengelim entegrasyon testi ile bir kısmının ise tümevarım entegrasyon testi kullanılarak

teste tabi tutulduğu test tipidir. Bu karma tipteki amaç ise bazı modülleri gruplara ayırabilirken diğer modülleri ise ayrı bir şekilde test edebilmektir.

Karma Entegrasyon Testi (Sandwich/Hybrid Integration Test): Yazılım geliştirme süreçlerinde, modüllerin bazıları tümdengelim (top-down) testine, diğerleri ise tümevarım (bottom-up) testine tabi tutulur. Bu test türü, modülleri gruplara ayırarak her birini farklı entegrasyon test yöntemleri ile test etmeye olanak tanır. Amaç, her modülün farklı seviyelerde test edilmesini sağlamak ve test sürecinde esneklik oluşturmaktır.

Bu testin uygulanabilirliğini artırmak için dikkate alınması gereken birkaç önemli husus vardır:

- Kullanıcı Deneyiminin Değerlendirilmesi: Karma entegrasyon testinin önemli bir yönü, kullanıcıların yazılımı nasıl deneyimleyeceklerini değerlendirme fırsatıdır. Bu test, yazılımın kullanıcı dostu ve etkileşimli olup olmadığını anlamaya yardımcı olur. Ayrıca, kullanıcının uygulama ile etkileşimi sırasında yaşadığı deneyimlerin optimize edilmesi için gerekli geri bildirimleri sunar.

- Etkileşim ve Veri Akışı Kontrolü: Yazılımın çeşitli bileşenleri arasındaki veri akışı, testlerin önemli bir parçasıdır. Kullanıcıların veri girişi sırasında oluşabilecek sorunları tespit etmek ve bunlara çözüm getirmek için bu test, etkileşimlerin ve veri akışının kontrolünü sağlar.

- Bileşenlerin Uyumu ve Hataların Erken Tespiti: Farklı modüller arasındaki etkileşim ve uyumluluk kontrol edilir. Bu sayede bileşenler arasındaki potansiyel uyumsuzluklar daha erken tespit edilerek, yazılımın tüm sistemle uyumlu çalışıp çalışmadığı değerlendirilir.

- Zaman ve Kaynak İhtiyacı: Karma entegrasyon testi, tüm sistemin entegrasyonu ve test edilmesi açısından zaman alıcı olabilir. Manuel testlerde ilave insan kaynağı gerektirir, bu da süreci uzatabilir.

- Öznel Değerlendirme: Kullanıcı deneyimi testleri öznel olabileceğinden, farklı testçiler arasında farklı sonuçlar elde edilebilir. Bu durum, test sonuçlarının doğruluğunu etkileyebilir.

- Sınırlı Kapsama ve Karmaşıklık: Karma entegrasyon testi, sadece yazılımın yüzeyini test eder, alt kodların karmaşıklığını değerlendirmez. Ayrıca, bileşenler arasındaki etkileşim arttıkça testler daha karmaşık hale gelir.

- Bileşen Bağımlılıkları: Eğer bir bileşen, diğerine bağımlıysa ve bu bağımlılık test sırasında sorun oluşturuyorsa, testler karmaşıklaşabilir ve başarısız olabilir.

Entegrasyon testinin avantajlarına aşağıda yer verilmektedir:

- Kullanıcı Deneyiminin Değerlendirilmesi: Kullanıcıların yazılım veya sistemi nasıl deneyimleyeceğini değerlendirmeye olanak tanır. Bu, kullanıcı dostu ve kolay kullanılabilir bir arayüzün sağlanmasına yardımcı olur.

- Etkileşim: Yazılımın farklı kullanıcı etkileşimlerine nasıl tepki verdiğini değerlendirir. Bu, kullanıcıların uygulamanın gerektiği gibi yanıt verdiğinden emin olmanıza yardımcı olur.

- Veri Akışının Kontrolü: Veri akışı ve veri girişi arayüz testleriyle kontrol edilir. Bu, yanlış veya eksik veri girişlerini tespit etmek için önemlidir.

- Bileşenlerin Uyumu: Farklı bileşenlerin bir araya geldiğinde uyumlu bir şekilde çalışıp çalışmadığını kontrol eder.

- Hataların Erken Tespiti: Yazılımın farklı bileşenlerini erken bir aşamada birleştirerek hataları erken tespit etmeye yardımcı olur.

- Sistem İyi Çalışırken Hataların Keşfedilmesi: Birim testler başarılı olabilirken bileşenlerin birleşmesi sonucu oluşabilecek hataları tespit etme fırsatı sunar.

Entegrasyon testinin dezavantajlarına aşağıda yer verilmektedir:

- Zaman ve Kaynak İhtiyacı: Tüm bileşenleri bir araya getirme ve test etme süreci zaman alabilir ve kaynaklar gerektirebilir. Manuel olarak yapılıyorsa, ilave zaman ve insan kaynağı gerektirir.

- Öznel Olabilir: Kullanıcı deneyimini değerlendirmek öznel bir süreç olabilir ve farklı testçiler arasında farklı sonuçlara yol açabilir.

- Sınırlı Kapsama: Yazılımın yüzeyini test eder, ancak altındaki kodun karmaşıklığını veya işleyişini değerlendiremez.

- Karmaşıklık: Farklı bileşenlerin etkileşimi nedeniyle karmaşık hale gelebilir.

- Bileşenlerin Bağımlılığı: Eğer bir bileşen diğerine bağımlıysa ve bu bağımlılık sorunluysa, testler karmaşık hale gelebilir.

c. Sistem Testi

Toplu olarak yeni veya değiştirilmiş bir sistem oluşturan değiştirilmiş programların, nesnelerin, veri tabanı şemasının vb. belirtilen sisteme uygunluğunun değerlendirilmesi için tasarlanmış eksiksiz bir entegre sistem üzerinde yapılan bir dizi testtir.

Sistem testi, yazılım geliştirme sürecinin önemli bir aşamasıdır ve tamamlanmış bir sistemin işlevselliğini, performansını ve uygunluğunu değerlendirmek için kullanılan bir test türüdür. Bu testin odak noktası, tüm sistemin belirtilen gereksinimlere uygunluğunu değerlendirmektir. Uygulamanın bir bütün olarak mimariyle ilgili iş, fonksiyonel, teknik ve işlevsel olmayan gereksinimlerinin onaylanmasına ve kontrol edilmesine yardımcı olur

Sistem testi, yazılımın tüm bileşenlerinin bir araya getirilmiş ve entegre edilmiş olduğu tamamlanmış bir sistem üzerinde gerçekleştirilir. Bu, sistemin gerçek kullanım koşullarına daha yakın bir şekilde test edilmesini sağlar. Sistem testi, değiştirilmiş veya yeni oluşturulmuş programların, nesnelerin, veritabanı şemalarının ve diğer bileşenlerin, belirli bir sisteme uygunluğunu değerlendirmek için kullanılır. Yazılım test döngüsüne uygun olarak, sistem testi kabul testinden önce ve entegrasyon testinden sonra yapılır. Sistem testi, sistemin eksiksiz bir şekilde test edildiği bir aşamadır. Bu, tüm işlevlerin ve gereksinimlerin yerine getirilip getirilmediğini kontrol etmeyi amaçlar. Sistemin performansını değerlendirmeyi içerir. Bu, sistem tarafından yönetilen iş yüküne dayanıklılığını, tepki sürelerini ve yük altında nasıl davrandığını test etmeyi içerir. Kullanıcılara canlı veya üretim ortamına az çok benzeyen etkili bir ortam sağlar.

Sistem testinin avantajlarına aşağıda yer verilmektedir:

- Birleşik Bileşenlerin Uyumu: Sistem testi, farklı bileşenlerin ve modüllerin bir araya geldiğinde uyumlu bir şekilde çalışıp çalışmadığını değerlendirir. Bu, tüm sistemin istenen şekilde bir araya geldiğinden emin olmayı sağlar.

- Kullanılabilirlik ve Güvenilirlik Kontrolü: Sistem testi, sistemin kullanılabilirlik, güvenilirlik ve istikrarını değerlendirir. Bu, kullanıcıların sistemi güvenle kullanabileceğinden emin olmayı amaçlar.

- Eksikliklerin ve Hataların Tespiti: Sistem testi, daha önceki test aşamalarında kaçırılmış veya gözden kaçırılmış hataları ve eksiklikleri tespit etmeye yardımcı olur.

- Performansı Değerlendirme: Sistem testi, sistemin performansını gerçek dünya koşullarında değerlendirmeyi sağlar. Bu, sistemin yük altında nasıl davrandığını anlamaya yardımcı olur.

- Kullanıcı Onayı İçin Temel: Sistem testi sonuçları, sistemin kullanıcılara sunulup sunulmayacağına dair önemli bilgiler sunar. Kullanıcıların sistemi kabul etmeleri için güvenilir bir temel oluşturur.

- Sistem testi, bir yazılım projesinin son aşamalarında uygulanan kritik bir test türüdür ve yazılımın tamamlanmış halini değerlendirme ve son onaylamayı sağlar.

d. Son Kullanıcı Testi

Uygulama aşamasında gerçeklesen, işletmenin kalite güvence metodolojisini uygulayan, son kullanıcı/müşteri isteklerine dayanan sistem testidir. Uygulamanın işlevsel yönüne odaklanır. Sistemin uygulamaya hazır olmasını ve dokümante edilmiş tüm gereksinimlerini karşılamasını sağlar. Canlı ortama benzer koşullarda güvenli bir test veya hazırlık ortamında gerçekleştirilir. Test adımları kullanıcı bakış açısı ile yazılarak, BT bölümü ve son kullanıcı ile gerçekleştirilir.

Bu testin amacı; yazılımı, iş gereksinimlerine göre doğrulamaktır. Yazılımın kullanıcı ve müşteri tarafından kabul edilip edilmeyeceğini belirlemek için test edilmesine olanak sağlar. Bu doğrulamalar iş gereksinimlerine aşina olan son kullanıcılar tarafından gerçekleştirilir. Fonksiyonel, sistem ve regresyon testleri gerçekleştirildikten sonra kullanıcı kabul testleri gerçekleştirilir. Yazılım yayınlanmadan gerçekleştirilen son testtir.

Son kullanıcı testinin avantajlarına aşağıda yer verilmektedir:

- Gerçek Kullanıcı Deneyimi: Gerçek kullanıcı deneyiminin değerlendirilmesine olanak tanır. Kullanıcıların yazılımı gerçek dünya koşullarında nasıl kullanacaklarının anlaşılmasına yardımcı olur.

- Gerçek Verilere Dayalı Test: Gerçek kullanıcılar, gerçek verilerle test yaparlar. Bu, yazılımın gerçek dünyadaki verileri nasıl işlediğini ve tepki verdiğinin gözlemlenmesini sağlar.

- Kullanıcı Geri Bildirimi: Son kullanıcılar, yazılımı kullanırken geri bildirim sağlarlar. Bu, yazılımın hatalarının ve kullanıcıların isteklerinin daha iyi anlaşılmasına yardımcı olur.

- Kullanıcı Kabulünü Sağlama: Yazılımın son kullanıcılar tarafından kabul edilip edilmediğini kesinleştirilmesine yardımcı olur. Kullanıcıların onayı, yazılımın kullanılabilirlik ve iş gereksinimlerini karşılayıp karşılamadığını gösterir.

- Geri Dönüş Zamanı: Yazılımın son halini test eder, bu nedenle hataların ve sorunların erken aşamalarda tespit edilmesine yardımcı olur ve geri dönüş süresini kısaltır.

Son kullanıcı testinin dezavantajlarına aşağıda yer verilmektedir:

- Zaman ve Kaynak İhtiyacı: Gerçek kullanıcıların katılımını gerektirir ve bu nedenle zaman ve kaynak yoğundur.

- Öznelllik: Son kullanıcıların deneyimleri öznel olabilir ve farklı kullanıcılar farklı sonuçlar verebilir.

- Beklenti Yönetimi Zorluğu: Kullanıcılar, yazılımdan farklı beklentilere sahip olabilirler. Bu, yazılımın herkesin beklentilerini karşılayamayabileceği anlamına gelir.

- Veri Gizliliği: Gerçek verilerle çalışıldığı için, son kullanıcı testlerinin veri gizliliği ve güvenliği konularına dikkat edilmesi gerekebilir.

- Uzak Kullanıcıların Katılımı: Uzak bölgelerdeki veya farklı coğrafyalardaki kullanıcıların katılımını sağlamak zor olabilir.

Diğer Testler:

a. Alfa ve Beta Testi

Alfa testi olarak adlandırılan ilk aşama, uygulama sisteminin ilk sürümlerinde yalnızca yazılımı geliştiren işletmedeki kullanıcılar veya yazılım geliştirici ekip tarafından gerçekleştirilir (sistem testleri). Geliştirmeden sonraki yazılım testinin ilk aşamasıdır. Yazılımın beta testine geçmesi için alfa testini geçmesi gerekir. Alfa testinin iki aşaması vardır. İlk aşama, geliştiriciler tarafından test edilmesidir. Temel amaç, hataları hızlı bir şekilde tespit etmektir. Yazılımın istenilen işlevselliği sağlamak için temel sorunların giderilmesi amaçlanır. Testin ikinci aşaması, sistemin kullanıcı tarafında olduğu gibi gerçek ortamda mükemmel bir şekilde çalışmasını sağlayan kalite güvence ekibi tarafından yapılmaktadır.

Beta testi (beta testing), gerçek kullanıcılar tarafından gerçek ortamda gerçekleştirilmektedir. Uygulamanın kullanıcılar tarafından test edildiği bir test çeşididir. Yazılımın gerçek dünyada nasıl çalıştığını daha iyi anlamak için önemlidir. Kullanıcılar için beta versiyonu yayınlanmaktadır. Kullanıcıların yazılımı gerçek kullanım senaryolarına göre test etmelerini sağlar. Ardından kullanıcılardan gelecek geri dönüşler doğrultusunda düzeltmeler yapılır. Kullanıcıların deneyimleri ve talepleri dikkate alınır. Beta testi, ürün arıza risklerini azaltır ve müşteri doğrulaması yoluyla ürünün kalitesinin artmasını sağlar.

Alfa testinin avantajlarına aşağıda yer verilmektedir:

- Geliştirici ekibinin hızlı geri bildirim almasını sağlar.

- Hataların erken tespiti ve düzeltilmesini sağlar. Böylece hem maliyet avantajı sağlar hem de kullanıcıların aynı hatalar ile karşılaşmasını önler.

- Uygulamanın bütünsel bir görünümünü ve geliştiricilerin başarıyı garantilemek için geliştirebilecekleri yolları sağlar.

- Kontrollü bir ortamda gerçekleştirilir.

Beta testinin avantajlarına aşağıda yer verilmektedir:

- Gerçek dünya koşullarında yazılımın performansının değerlendirilmesine olanak sağlar.

- Kullanıcı geri bildirimlerine dayalı iyileştirmeler yapılır.

- Ürün arıza risklerini azaltır.

- Müşteri doğrulaması ile ürünün kalitesini artırır.

Sonuç olarak, alfa testi bir uygulamanın genel kullanılabilirliği ve işlevselliği etrafında dönerken, beta testi kararlılık, güvenilirlik ve güvenliğe daha fazla vurgu yapar. Bu kontroller, programın hem beklenen hem de beklenmeyen girdileri nasıl ele aldığını görmeyi içerir. Ayrıca alfa testi ve beta testi, yazılımın farklı aşamalarında hataların tespiti, geliştirme ve kullanıcı geri bildirimlerine dayalı iyileştirmelerin yapılması için önemli test türleridir. Alfa testi daha kontrollü bir ortamda gerçekleştirilirken, beta testi gerçek kullanıcılar tarafından gerçek dünya koşullarında yapılır. Bu iki aşama, yazılımın daha sağlam ve kullanılabilir hale gelmesine katkı sağlar.

b. Pilot Testi

Sistemin bir bileşenini veya tüm sistemi gerçek zamanlı bir çalışma koşulu altında doğrulayan kavram kanıtı gibi bir sistemin spesifik ve önceden belirlenmiş yönlerine odaklanan bir yazılım testidir. Amacı bir araştırma projesinin fizibilitesini, zamanını, maliyetini, riskini ve performansını değerlendirmektir. Bu test tam olarak UAT ve üretim arasında yapılır.

Pilot testinde, seçilen bir grup son kullanıcı test edilen sistemi dener ve sistemin tam olarak konuşlandırılmasından önce geri bildirim sağlar. Diğer bir deyişle, aşağıdaki kullanılabilirlik testi için kostümlü prova yapmak anlamına gelir.

Pilot testi, sistemdeki hataların erken tespitine yardımcı olur.

Pilot testi, sürekli ve düzenli kullanıma karşı test etmek için müşteri sahasına (veya kullanıcı simülasyonlu bir ortama) bir sistem kurmakla ilgilidir. En yaygın test yöntemi, zayıf alanlarını bulmak için sistemi sürekli olarak test etmektir. Bu zayıflıklar daha sonra geliştirme ekibine hata raporları olarak geri gönderilir ve bu hatalar sistemin bir sonraki yapısında giderilir. Bu süreçte bazen kabul testleri de uyumluluk testinin parçası olarak dahil edilir. Bu, eskisinin yerini alacak bir sistem geliştirildiğinde ortaya çıkar.

Yazılım mühendisliğinde pilot testi, ürün veya hizmetin potansiyel bir pazarı olup olmadığı gibi soruya cevap verecektir.

Bu test aşağıda belirtilen nedenlerden dolayı önemlidir:

- Test için kullanılan yazılım ve prosedürde hata ayıklama, tam ölçekli uygulama için ürün hazırlığını kontrol etme, zaman ve kaynak tahsisinde daha iyi karar verme, hedef kitlesinin programa tepkisini ölçme fırsatı verir; başarı ölçümü ekibe kullanılabilirlik testi için kullanacakları etkinlikleri uygulama şansı verir.

- Gerçek zamanlı çalışma koşulları altında sistemin bir bileşenini veya tüm sistemi doğrulamaktır.

- Tam olarak UAT ve üretim arasında yapılır.

- Ürünün tam ölçekli uygulamaya hazır olup olmadığını kontrol etmeye yardımcı olur.

Pilot testi için önemli hususlara aşağıda yer verilmektedir:

- Hedef Kitlenin Tanımlanması: Pilot testi yaparken, hedef kitleyi dikkate almak önemlidir. Hangi kullanıcılar veya müşteri grupları hedefleniyor? Bu grupların ihtiyaçları, beklentileri ve geri bildirimleri, pilot testinin başarısını etkileyebilir.

- Sistemin Özgünlüğü: Genellikle yeni veya özgün bir sistem veya yazılımın değerlendirilmesi için kullanılır. Bu nedenle, sistemin özgün özellikleri ve benzersizliği üzerinde durulmalıdır.

- Gerçek Ortam Simülasyonu: Gerçek dünya koşullarını simüle etmeyi amaçlar. Bu, sistemin gerçek kullanım senaryolarına ve çevresel faktörlere nasıl tepki verdiğini daha iyi anlamak için önemlidir.

- Hata Ayıklama ve İyileştirme: Hataların erken tespit edilmesine yardımcı olur. Bu hatalar daha sonra geliştirme ekibi tarafından düzeltilir ve sistemin istenilen performansı sunması sağlanır.

- Kullanılabilirlik ve Etkinlik: Sistemin kullanılabilirliğini ve etkinliğini değerlendirmeyi amaçlar. Kullanıcıların sistemi rahatça kullanabilmesi ve işlerini daha iyi yapabilmesi gerekmektedir.

- Maliyet ve Risk Değerlendirmesi: Proje maliyetlerini ve risklerini daha iyi anlamayı sağlar. Bu, proje yönetimi açısından önemlidir ve gerekli düzeltmelerin yapılmasına yardımcı olur.

- Prototip Oluşturma: Bazı durumlarda, pilot testi için özel bir prototip oluşturulur. Bu prototip, sistemin belirli yönlerini veya özelliklerini test etmek için kullanılır ve gerçek sistemin geliştirilmesi öncesinde bir ön izleme sunar.

- Pazar Araştırması: Pilot testi aynı zamanda pazar araştırması için bir fırsat sunar. Ürün veya hizmetin hedef pazarda nasıl karşılandığını değerlendirmek ve pazarın tepkisini ölçmek için kullanılabilir.

Sonuç olarak, pilot testi, yazılım geliştirme veya yeni ürün/hizmet lansmanı süreçlerinde önemli bir aşamadır. Bu aşamada sistemin özgünlüğü, kullanılabilirliği, hata ayıklama süreci ve hedef kitlenin gereksinimleri dikkate alınmalıdır. Pilot testi, sistemin başarılı bir şekilde pazara sürülmesi için önemli bir adımdır ve geliştirme sürecine değerli geri bildirimler sağlar.

c. Beyaz Kutu Testi

Beyaz kutu testi, birim testi ve entegrasyon testi dahil olmak üzere birçok farklı yazılım testi türünü içeren bir şemsiye terimdir. Beyaz kutu testi, bir programın/modülün beklenen davranışını doğrulamak için bu program/modülün uygulanmasının ve kod aralıkları bilgisinin kullanıldığı bir test yaklaşımıdır. Bu test tasarım tekniği veri akışlarına, kontrol akışlarına, ifade kapsama, dal kapsama gibi konulara odaklanır. Beyaz kutu testleri aynı zamanda saydam (cam) kutu testi olarak da bilinir. Yazılımın dış kabuğunu, kutunun iç işleyişini görmeyi sembolize eder. Aynı şekilde, kara kutu testinde yer alan kara kutu, kutunun, yazılım içeriğinin görülmediğini ve kapalı olduğunu temsil eder. Son kullanıcı sistemin içini görmez. İşlevselliği ile ilgilenir. Beyaz kutu testi, en yaygın olarak birim testi yapılırken ve bazen entegrasyon testi sırasında gerçekleştirilir.

Beyaz kutu testinin avantajlarına aşağıda yer verilmektedir:

- Detaylı Test İmkanı: Kodun iç yapısını inceleyerek detaylı bir test yapma imkanı sağlar. Bu, kodun her bir bileşenini ve akışını kontrol etmek için kullanışlıdır.

- Kod Kapsamını Değerlendirme: Kod kapsamını değerlendirmek için kullanılabilir. Bu sayede, hangi kod bloklarının test edildiği ve hangilerinin test edilmediği daha iyi anlaşılabilir.

- Hata Ayıklama Kolaylığı: Kodun iç yapısını görmek, hataları daha kolay tespit etmeyi sağlar. Hatanın kod tabanında tam olarak nerede olduğunu bulmayı kolaylaştırarak ayıklama sürecini hızlandırır.

- Kod İyileştirme: Yazılım geliştiricilerine kodlarını iyileştirmek için rehberlik edebilir. Kod kalitesini artırmak ve verimliliği artırmak için geri bildirim sağlar.

- Erişim Kontrolü: Belirli kod bloklarını ve fonksiyonları izole etmek ve test etmek için erişim kontrolü sağlar. Bu, belirli bileşenlerin beklenen şekilde çalıştığını doğrulamayı kolaylaştırır.

- Gizli Hataların Tespiti: Kodun derinliklerinde gizli hataları ortaya çıkarabilir. Kod içindeki potansiyel hataları tespit etmek, yazılımın daha güvenilir olmasını sağlar.

Beyaz kutu testinin dezavantajlarına aşağıda yer verilmektedir:

- Zaman ve Kaynak İhtiyacı: Kodun iç yapısının ayrıntılı bir şekilde incelenmesini gerektirir, bu nedenle zaman ve kaynak yoğundur.

- Kod Değişikliklerine Duyarlılık: Kodun iç yapısındaki değişiklikler, beyaz kutu testlerinin güncellenmesini gerektirebilir. Bu, bakım maliyetlerini artırabilir.

- Kod Bilgisi Gerektirir: Gerçekleştirmek için test ekibinin yazılım kodunu anlaması ve bu kodu incelemesi gereklidir. Bu nedenle, belirli bir uzmanlık seviyesine ihtiyaç duyulabilir.

- Eksik Kapsam: Kodun belirli kısımlarını test eder, ancak tüm olası senaryoları kapsamayabilir. Bu nedenle, bazı hatalar kaçırılabilir.

d. Kara Kutu Testi

Kara kutu testi, uygulamanın veya ürünün işlevselliğine odaklanan ve kod aralıkları bilgisi gerektirmeyen bir test yaklaşımıdır. Bu testlerdeki amaç, gereksinimleri karşılayan çıktıların alınıp alınmadığının ölçümlenmesidir. Sistemden alınması beklenen çıktılar kadar beklenmeyen çıktılar da test edilmelidir. Bu test tasarım tekniği, test uzmanlarının sıklıkla kullandığı bir yöntemdir. Kara kutu testi; birim, entegrasyon, sistem ve kullanıcı kabul testi gibi testlerin her seviyesinde kullanılabilir.

Yazılım test tasarım tekniklerinden olan beyaz kutu testleri geliştirilen yazılımın içyapısı ve iş akışıyla ilgilenirken kara kutu testleri sistemin işlevselliği ile ilgilenir. Kara kutu testleri son kullanıcı perspektifinden test etmeyi içerir. Dolayısıyla kod içeriği ve işleyişin bilinmesine gerek yoktur.

Kara kutu testinin avantajlarına aşağıda yer verilmektedir:

Bağımsızlık: Yazılımın iç yapısını veya kodunu incelemeden gerçekleştirildiği için, test ekibi ve kullanıcılar için bağımsız bir bakış açısı sunar.

Kullanıcı Odaklılık: Kullanıcıların beklediği işlevselliği değerlendirmeye odaklanır. Bu, son kullanıcıların deneyimini doğrudan yansıtır.

Gereksinim Uyumluluğu: Belirtilen gereksinimlere uygunluğu değerlendirir. Bu, yazılımın gereksinimlere uygun olarak çalışmasını sağlar.

Testin Tekrarlanabilirliği: Test senaryolarını tekrar kullanarak testi tekrar etmeyi kolaylaştırır. Bu, yazılım değişiklikleri yapıldığında testlerin hızla tekrarlanabilmesine olanak tanır.

Kara kutu testinin dezavantajlarına aşağıda yer verilmektedir:

Tam Kapsam Güvencesi: Yazılımın iç yapısını incelemediği için, belirli kod kısımlarını veya iç işleyişi test etmez. Bu nedenle, bazı hatalar bu testlerde yakalanamayabilir.

Performans Sorunları: Yazılımın performansını veya verimliliğini doğrudan ölçemez. Bu tür sorunlar için ayrı testler gerekebilir.

Hata Ayıklama Zorluğu: Tespit edilen hataların kaynağını belirlemek ve düzeltmek daha zor olabilir. Çünkü testler, kodun iç yapısına erişim sağlamaz.

Güvenlik Testi Zayıflığı: Güvenlik açıkları veya zayıflıkları üzerine spesifik olarak odaklanmaz. Bu tür testler için ek güvenlik testleri gerekebilir.

e. Fonksiyonel/Doğrulama Testi

Fonksiyonel/doğrulama testi, oluşturulan yazılımın müşteri gereksinimlerine göre izlenebilmesini sağlamak için sistemin işlevselliğini ayrıntılı gereksinimlere karşı test eder.

Yazılımın tüm gereksinimlere uygun olduğundan emin olmak için test edildiği, yazılım geliştirme içinde kullanılan bir yazılım test sürecidir. İşlevsel test, yazılımı, işlevsel gereksinimlerinde belirtilen tüm gerekli özelliklere sahip olduğundan emin olmak için kontrol etmenin bir yoludur.

Aşağıdakileri kontrol etmek için genellikle işlevsel test kullanılır:

- Temel İşlevsellik

- Temel Kullanılabilirlik

- Ulaşılabilirlik

- Hata Koşulları

Fonksiyonel/Doğrulama testinin avantajlarına aşağıda yer verilmektedir:

- Müşteri Gereksinimlerine Uyum: Yazılımın müşteri gereksinimlerine uygun olup olmadığını doğrulamak için kullanılır. Bu, müşteri memnuniyetini artırır.

- Temel İşlevselliğin Testi: Temel işlevselliği kontrol etmek için kullanılır. Yazılımın en önemli ve temel özelliklerinin doğru çalıştığını doğrulamak için kullanışlıdır.

- Kullanılabilirlik Kontrolü: Yazılımın kullanıcı dostu olduğunu ve kullanılabilir olduğunu belirlemek için kullanılır.

- Hata Koşullarının Testi: Hata koşulları ve hata mesajlarının doğru şekilde işlendiğini ve kullanıcıya uygun bir şekilde sunulduğunu kontrol eder.

Fonksiyonel/Doğrulama testinin dezavantajlarına aşağıda yer verilmektedir:

- Sınırlı Kapsam: Yazılımın temel işlevselliğini test eder, ancak tüm olası senaryoları kapsamayabilir. Bu nedenle, bazı hatalar kaçırılabilir.

- Performans Testi Eksikliği: Yazılımın performansını veya yük altındaki davranışını ölçmez. Bu tür testler için ayrı bir performans testi gerekebilir.

- Güvenlik Testi Eksikliği: Güvenlik açıkları veya zayıflıkları üzerine özel olarak odaklanmaz. Bu tür testler için ek güvenlik testleri gerekebilir.

- Hata Kaynaklarının Belirlenmesi: Hata kaynaklarını belirlemekte sınırlıdır ve hataların kökenini tespit etmek ve düzeltmek daha fazla analiz gerektirebilir.

Fonksiyonel/Doğrulama testi, yazılımın temel işlevselliğini ve müşteri gereksinimlerini karşılayıp karşılamadığını belirlemek için önemli bir test türüdür. Ancak, eksiklikleri ve sınırlamaları nedeniyle diğer test türleriyle bir araya getirilerek daha kapsamlı bir test stratejisi oluşturulabilir.

f. Regresyon Testi

Regresyon testi, canlıda çalışan uygulama/kod vb. üzerindeki değişikliklerin veya düzeltmelerin yeni hatalar getirmediğinden emin olmak için bir test senaryosunun veya test planının bir kısmını yeniden koşma işlemidir. Bu değişiklik/düzeltmeler yeni bir fonksiyon, hata çözümü ya da performans geliştirmesi olabilir. Regresyon testleri genellikle değişiklikler son aşamaya geldiğinde ve yazılımın yeni sürümü yayınlanmadan önce gerçekleştirilir. Regresyon testlerinin öncelikli amacı, uygulamanın kritik alanlarının hala beklendiği gibi çalıştığını kontrol etmektir. Regresyon testleri:

- Yazılımın değişiklik sonrasında, son kalitesinin kontrol edilmesini,

- Daha önce çıkan hataların düzeldiğinin kontrolünü,

- Yazılım ekibinin ürün hakkında güveninin artmasını,

sağlar.

Yazılım hataları veya gerilemelerin (performans düşüklüğü), fonksiyonel ve fonksiyonel olmayan geliştirmelerin, sistemin bazı alanlarına yapılan yamaların, yapılandırma değişiklilerinin regresyon testi ile etkileri izlenebilir.

Regresyon testinin avantajlarına aşağıda yer verilmektedir:

- Hata Kontrolü: Daha önce bulunan hataların düzeltildiğini ve yeni hataların eklenmediğini doğrulamak için kullanılır.

- Değişikliklerin İzlenmesi: Yazılımın herhangi bir değişiklik veya güncelleme sonrasındaki davranışını izlemek ve değerlendirmek için kullanışlıdır.

- Güven Artışı: Yazılım ekibinin ürüne olan güvenini artırır. Değişikliklerin veya düzeltmelerin yazılımın genel performansını olumsuz etkilemediğini gösterir.

- Yeniden Kullanılabilirlik: Regresyon test senaryoları, yazılımın farklı sürümlerinde tekrar kullanılabilir. Bu, test sürecinin verimliliğini artırır.

Regresyon testinin dezavantajlarına aşağıda yer verilmektedir:

- Zaman ve Kaynak İhtiyacı: Büyük yazılım projelerinde regresyon testleri zaman alıcı olabilir. Her güncelleme sonrasında tüm test senaryolarını yeniden çalıştırmak gerekebilir.

- Kapsamlı Test Senaryoları: Kapsamlı bir test senaryosu oluşturmak zor olabilir. Hangi senaryoların dahil edilip hangilerinin atlanacağını belirlemek önemlidir.

- Eksik Testler: Mevcut test senaryolarını yeniden çalıştırmayı amaçlar, ancak yeni eklenen özellikleri veya değişiklikleri kapsamayabilir. Bu nedenle, yeni işlevleri test etmek için ayrı testler gerekebilir.

- Otomasyon Zorluğu: Otomatize etmek bazen zor olabilir. Özellikle kullanıcı arayüzüne dayalı testler, otomatize edilmesi zor olan testler arasında yer alabilir.

Regresyon testi, günümüzde birçok yazılım yinemeli geliştirim süreci ile yapıldığından önemlidir. Bu süreçte, her bir döngüde yeni bazı işlevler eklenerek kısa döngüler kullanıldığından yeni eklenen işlevlerin var olan işlevleri bozmadığından emin olmak için, her bir döngüde bu testi uygulamak anlamlıdır. Ayrıca bu test, yazılımın değişiklikler sonrasındaki güvenilirliğini ve bütünlüğünü sağlamak için de önemlidir. Ancak bu testlerin planlanması, yönetimi ve otomasyonu dikkatle ele alınmalıdır.

g. Paralel Test

Paralel test, test verisini iki sisteme, değiştirilmiş sisteme ve alternatif bir sisteme (muhtemelen orijinal sistem), besleme ve uygulamanın iki sürümü arasındaki sonuçları karşılaştırma sürecidir. Bu test, test yürütme süresini azaltmak için bir uygulamanın birden çok sürümünün veya alt bileşeninin aynı anda farklı sistemlerde aynı girişle test edildiği bir yazılım test türüdür. Bu testin amacı, eski sürüm ile yeni sürümün aynı mı yoksa farklı mı davrandığını bulmak ve yeni sürümün daha verimli olup olmadığını sağlamaktır. Paralel testin yapılma nedenleri:

- Uygulamanın yeni sürümünün doğru çalıştığından emin olmak,

- Yeni ve eski sürüm arasındaki tutarlılıkların aynı olduğundan emin olmak,

- İki sürüm arasındaki veri formatının değişip değişmediğini kontrol etmek,

- Yeni uygulamanın bütünlüğünü kontrol etmek,

- Test süresini kısaltmak ve test verimliliğini artırmak.

Paralel testin avantajlarına aşağıda yer verilmektedir:

- Doğruluk Kontrolü: Yeni bir sürümün veya sistem değişikliğinin doğru çalıştığından ve istenen sonuçları ürettiğinden emin olmak için kullanılır.

- Tutarlılık Kontrolü: Yeni ve eski sürüm veya sistem arasındaki tutarlılığı değerlendirmek için kullanılır. Verilerin ve sonuçların aynı olup olmadığını kontrol eder.

- Veri Formatı Kontrolü: Veri formatının değişip değişmediğini belirlemek ve kontrol etmek için kullanılır.

- Bütünlük Kontrolü: Yeni uygulamanın veya sürümün bütünlüğünü değerlendirmek için kullanılır. Verilerin eksik veya bozuk olup olmadığını kontrol eder.

Paralel testin dezavantajlarına aşağıda yer verilmektedir:

- Zaman ve Kaynak İhtiyacı: Birden fazla sistemde veya sürümde aynı anda çalıştığı için zaman ve kaynak yoğundur.

- Yönetim Zorluğu: Yönetimi karmaşık olabilir. Her iki sistem veya sürüm için uygun test senaryolarının tasarlanması ve uygulanması gereklidir.

- Otomasyon Zorluğu: Otomatize edilmesi, yazılım test otomasyonu için daha fazla karmaşıklık getirebilir.

- Sonuçların Karşılaştırılması: Test sonuçlarının karşılaştırılması ve analizi önemlidir, ancak bu da ek iş yükü gerektirebilir.

h. Uyumluluk Testi

Uyumluluk testi, yeni veya değiştirilmiş sistemin, var olan sistemleri olumsuz yönde etkilemeden hedef ortamında çalışabileceğini onaylamak için yapılan testtir. Yazılan kodun uyumluluğunu kontrol eder. Donanım/yazılım/işletim sistemi/ağ ortamında ne kadar iyi performansa sahip olduğunu tespit eder ve doğrular. Bu testin hedefleri aşağıdaki gibidir:

- Geliştirme ve bakım sürecinin öngörülen metodolojiyi karşıladığının belirlenmesi,

- Geliştirmenin her aşamasında oluşan çıktıların standartlara, prosedürler ve yönergelere uygun olup olmadığını garanti etme,

- Eksiksiz ve makul olup olmadığını kontrol etmek için projenin belgeleri değerlendirme.

Uyumluluk testinin avantajlarına aşağıda yer verilmektedir:

- Hedef Ortamda Güvenilirlik: Yazılımın veya sistemin hedef ortamda güvenilir bir şekilde çalıştığını doğrular. Bu, olası hataları önceden tespit etmeye yardımcı olur.

- Yazılım ve Donanım Uyumluluğu: Yazılımın belirli donanım, işletim sistemi veya ağ ortamlarıyla uyumlu olduğunu kontrol eder. Bu, farklı platformlarda sorunsuz çalışabilirliği sağlar.

- Metodoloji Uygunluğu: Geliştirme ve bakım süreçlerinin öngörülen metodolojiyi karşıladığını belirler. Standartlara, prosedürlere ve yönergelerin takip edildiğini garanti eder.

- Belge Kontrolü: Projenin belgelerini değerlendirerek eksik veya uygun olmayan bilgileri tespit eder.

Uyumluluk testinin dezavantajlarına aşağıda yer verilmektedir:

- Kaynak Yoğunluğu: Farklı ortamlarda test yapmayı gerektirebilir. Bu, kaynakların ve zamanın yoğun bir şekilde kullanılmasına neden olabilir.

- Kapsamlı Test Senaryoları: Farklı ortamların ve uyumlu bileşenlerin tüm senaryolarını kapsayan uyumluluk test senaryolarını oluşturmak zor olabilir.

- Test Ortamlarının Yönetimi: Farklı test ortamlarının yönetimi ve konfigürasyonu karmaşık olabilir.

- Uyumluluk Sorunlarının Kararlılık Sorunlarına Neden Olması: Uyumluluk sorunları, yazılımın kararlılığını olumsuz etkileyebilir ve hatalara yol açabilir.

Veri Bütünlüğü Testi:

Veri bütünlüğü testi**,** veritabanlarında veya veri ambarlarında tutulan verilerin doğruluğunu, bütünlüğünü, tutarlılığını, işlevselliğini, kalitesini ve yetkilendirmesini inceleyen manuel/otomatik bir dizi temel testtir.

Veri bütünlüğü testinin avantajlarına aşağıda yer verilmektedir:

- Veri Kalitesi Garantisi: Veritabanındaki verilerin doğruluğunu ve kalitesini sağlar. Bu, yanlış veya eksik veri kayıtlarını tespit etmeye yardımcı olur.

- Tutarlılık Kontrolü: İlişkisel bütünlük testleri, veriler arasındaki ilişkilerin ve referansların doğru olduğunu kontrol eder. Bu, veritabanı bütünlüğünü korumaya yardımcı olur.

- Yetkilendirme Kontrolü: Verilere erişim yetkilerini doğrular. Verilere izinsiz erişimi engeller.

- Veritabanı Performansını Artırma: Gereksiz veya yavaş veritabanı sorgularını belirleyerek veritabanı performansını artırabilir.

Veri bütünlüğü testinin dezavantajlarına aşağıda yer verilmektedir:

- Kaynak Yoğunluğu: Veritabanı büyüklüğüne ve karmaşıklığına bağlı olarak zaman ve kaynak yoğun olabilir.

- Test Senaryolarının Hazırlanması: Test senaryolarının hazırlanması ve uygulanması zaman alıcı olabilir.

- Otomasyon Zorluğu: Otomatize etmek zor olabilir, çünkü veri manipülasyonu ve doğrulama gerekebilir.

- Veri Güncellemeleri ile Uyumsuzluk: Veritabanı güncellemeleri veya yapısal değişiklikler, mevcut test senaryolarının uyumsuz hale gelmesine neden olabilir.

Veri bütünlüğü testleri, verilerin doğruluğunu ve güvenilirliğini korumak için kritik öneme sahiptir. Ancak bu testlerin planlanması ve yönetilmesi, veritabanının boyutu ve karmaşıklığına bağlı olarak zorlu olabilir.

Aşağıda bu testlerin yaygın türlerine yer verilmektedir:

İlişkisel bütünlük testleri - Veri öğesi ve kayıt tabanlı seviyelerde yapılan veri doğrulama rutinleridir. İlişkisel bütünlük testleri, veri tabanındaki veri öğeleri ve kayıtlar arasındaki ilişkilerin doğru ve uygun olduğunu doğrulamak için yapılır. Örneğin, bir müşteri ve sipariş veri tabanında, her müşterinin bir benzersiz kimlik numarasına sahip olduğunu ve her siparişin bir müşteriye ait olduğunu doğrulayabilirler. Bu test yöntemi, verilerin doğru ve tutarlı olduğundan emin olmayı sağlar, veri tabanı bütünlüğünü korur. Test senaryolarının hazırlanması ve yönetilmesi zaman alıcı olabilir.

Bilgi tutarlılığı testleri - VTYS (DBMS) tarafından korunması gereken bir veri tabanının farklı tablolarındaki varlıklar arasındaki varlık ilişkilerinin tanımlanmasıdır. Bilgi tutarlılığı testleri, veri tabanı yönetim sistemi (DBMS) tarafından korunması gereken bir veri tabanının farklı tablolarındaki varlıklar arasındaki ilişkileri ve tutarlılığı incelemek için kullanılır. Farklı tablolar arasındaki veri uyumsuzluklarını tespit etmeye yardımcı olur, veri tabanının bütünlüğünü ve veri kalitesini artırır. Örneğin, bir eğitim kurumu veri tabanında, öğrencilerin kayıtlarının ders programlarıyla uyumlu olduğunu ve eksik veya çakışan ders kayıtlarının olmadığını doğrulayabilirler.

Bulut Tabanlı Veri Bütünlüğü Testleri - Bulut tabanlı veri yönetimi ve depolama çözümleri, verilerin sürekli olarak dağıtılması ve ölçeklendirilmesi ile ilgili karmaşıklıklar getirir. Bulut ortamında veri bütünlüğü sağlamak, verilerin doğru ve güvenli bir şekilde saklanıp işlenmesini garanti etmek için, bulut servis sağlayıcılarının verilerin bütünlüğünü nasıl koruduğunu ve izlediğini test etmek önemlidir. Bu, özellikle veri yedekleme, şifreleme ve veri kurtarma süreçlerinde kritik hale gelir.

Büyük Veri (Big Data) Bütünlüğü Testleri - Günümüzde büyük veri sistemlerinde veri bütünlüğü sağlamak daha karmaşık hale gelmiştir. Büyük veri setleri genellikle farklı kaynaklardan gelen yapılandırılmamış veya yarı yapılandırılmış veriler içerir. Bu nedenle, büyük veri işleme ve analitik platformlarında, veri kalitesini ve bütünlüğünü sağlamak için verilerin doğru şekilde depolandığı, işlendiği ve raporlama için kullanılmadan önce doğru hale getirildiği test edilmelidir. Bu süreç, özellikle Hadoop veya Spark gibi büyük veri platformlarında veri akışlarının doğru yönetilmesiyle ilgili testleri içerir.

Veri Gizliliği ve Güvenliği Testleri - Veri güvenliği ve gizliliği, günümüz veri tabanı testlerinin ayrılmaz bir parçasıdır. Özellikle kişisel verilerin (PII) veya sağlık verilerinin bulunduğu sistemlerde, veri bütünlüğü testleri ile birlikte, verilerin doğru bir şekilde şifrelenip şifresinin çözülmediği, yalnızca yetkilendirilmiş kullanıcıların verilere erişimi sağlandığı ve verilerin güvenliğini koruyan protokoller kullanıldığı doğrulanmalıdır.

Veri Senkronizasyonu Testleri - Modern uygulamalarda, veriler genellikle birden fazla cihaz veya sistem arasında senkronize edilir. Veri bütünlüğü testi, verilerin bu sistemler arasında doğru şekilde

senkronize olup olmadığını kontrol etmelidir. Örneğin, bulut tabanlı bir hizmet ile yerel bir veri tabanı arasındaki veri senkronizasyonunun düzgün yapılıp yapılmadığı test edilmelidir.

Veri Anomalisi Tespiti ve Makine Öğrenmesi (ML) Kullanımı - Veri bütünlüğü testlerinde geleneksel yöntemlerin yanı sıra, anormal veri girişlerini tespit etmek için makine öğrenmesi algoritmaları kullanılabilir. Makine öğrenmesi, veritabanındaki tutarsızlıkları, hataları ve olası veri bozulmalarını otomatik olarak tespit edebilecek bir araç olarak kullanılabilir. Veri anomali tespiti, özellikle büyük veri setlerinde hızlı ve doğru hata tespiti için etkili bir yöntem olabilir.

Blockchain Teknolojisi ile Veri Bütünlüğü - Blockchain, verilerin değiştirilemez ve şeffaf bir şekilde saklanmasına olanak tanır. Veri tabanı testlerinde blockchain teknolojisinin kullanılması, özellikle finansal verilerin ve diğer hassas verilerin güvenliğini sağlamak için etkili olabilir. Blockchain'in şeffaflık özellikleri, veri bütünlüğünü sağlamada önemli bir rol oynar ve verilerin doğruluğunu kontrol etmek için yeni bir yaklaşım sunar.

Yazılım Testi:

Yazılım testleri, kullanıcı gereksinimlerinin doğrulanmış olduğunu, sistemin beklendiği gibi çalıştığını ve dahili kontrollerin amaçlandığı şekilde çalıştığını belirler. Bu sayede yazılımdaki hatalar bulunup düzeltilebilir ve gereksinimlere uygun hale getirilebilir. Yazılım testi ise bir yazılımın sonsuz sayıdaki çalışma alanından, sınırlı sayıda ve uygun şekilde seçilmiş testler ile beklenen davranışlarını karşılamaya yönelik, dinamik olarak yapılan doğrulama faaliyetlerini kapsamaktadır.

Yazılım testi, bir anlamda maliyet, zaman, kalite kıstaslarının optimize edilmesidir.

Bu teste iki ana yaklaşım bulunur:

1. Aşağıdan-yukarıya/Tümevarım (bottom-up approach)- Her bir birimin testi ile başlar. Birim testleri başarıyla tamamlanan alt düzey modüller birbirleriyle entegre edilir. Bu entegrasyondan sonra alt modüller ile ilgili entegrasyon testleri yapılır. Bu testlerin başarılı şekilde sonlandırılmasından sonra bir üst katman modülleri sistemi entegre edilir ve 44 gerekli entegrasyon testleri yapılır. Bu entegrasyon tüm sistem inşa edilinceye ve entegrasyon testleri başarıyla sonuçlanıncaya kadar devam eder. Bu yaklaşımda temel şart, entegre edilecek modüllerin birim testlerinin başarıyla tamamlanmış olmasıdır. Bu yaklaşım ile hedeflenen sistemin yüzdelik olarak ne kadarının tamamlandığı ve test edildiği kolay bir şekilde belirlenebilir.

2. Yukarıdan-aşağıya/Tümdengelim (top-down approach)**-** Öncelikle sistem için en üst modülünün (en dış modül, genellikle kullanıcı grafik arayüz modülü) test edilmesiyle başlar. Bu modülün ihtiyaç duyduğu alt modüllerinin koçanları kullanılır. Bu yaklaşımda birçok koçan yazılması gerekir. En üst modül başarılı şekilde test edildikten sonra bir alt düzey modüller sistemle bütünleştirilir (sce.uhcl.edu). Bu entegrasyondan sonra gerekli alt düzey tümleştirme testleri yapılır. Bu durum en alt düzey modüller entegre edilinceye ve entegrasyon testleri başarıyla sonuçlanıncaya kadar devam eder. Bu yaklaşımda entegrasyon testleri kara kutu testleri olarak başlar ve ilerledikçe saydam kutu testlerine döner.

Aşağıdan yukarı yaklaşım ile yukarıdan aşağı yaklaşım arasındaki farklara aşağıda yer verilmektedir:

Aşağıdan Yukarı Yaklaşım:

- Başlangıç Seviyesi: Bu yaklaşım alt düzey modüllerden (birimlerden) başlar ve sistemin daha üst seviyelerine doğru ilerler.

- Modül Odaklı: Odak, alt düzey modüllerin doğru bir şekilde çalıştığına ve test edildiğine dayanır.

- Birim Testleri İlk Olarak: Birim testleri, alt düzey modüllerin test edilmesi ile başlar ve bu modüllerin işlevselliğini doğrular.

- Entegrasyon Gevşek: Entegrasyon testleri, alt düzey modüllerin sisteme entegrasyonu ile ilgilenir ve daha sonra bu modüllerin birleşimini kontrol eder.

- Alt Düzey Modüller İlk Olarak: Entegrasyon testleri, alt düzey modüllerin entegrasyonunun başarılı bir şekilde tamamlanmasını bekler.

- Uygulama Üst Düzeyde Tamamlanır: Sistem, alt düzey modüllerden başlayarak üst düzey modüllerin birleşimiyle tamamlanır.

- Kara Kutu Testlerle Başlar, Saydam Kutu Testlere Dönüşebilir: Entegrasyon testleri genellikle kara kutu testlerle başlar ve ilerledikçe saydam kutu testlere dönebilir.

- Özellikle Detaylı: Bu yaklaşım alt düzey modüllerin detaylarına odaklanır ve alt modüllerin birim testlerinin başarılı bir şekilde tamamlanmış olmasını gerektirir.

Yukarıdan Aşağı Yaklaşım:

- Başlangıç Seviyesi: Bu yaklaşım üst düzey modül veya bileşenlerden başlar ve daha alt düzeylere doğru ilerler.

- Sistem Odaklı: Odak, sistemin en üst düzey işlevselliğini doğrulamaya ve üst seviye gereksinimleri karşılamaya dayanır.

- En Üst Düzey Modüller İlk Olarak: En üst düzey modüller veya sistemin başladığı yer, ilk olarak test edilir.

- Entegrasyon Gevşek: Entegrasyon testleri, daha üst düzey modüllerin alt düzey modülleri nasıl kullandığını kontrol eder ve daha sonra bu modüllerin birleşimini test eder.

- Üst Düzey Modüller İlk Olarak: Entegrasyon testleri, daha üst düzey modüllerin entegrasyonunun başarılı bir şekilde tamamlanmasını bekler.

- Uygulama Alt Düzeyde Tamamlanır: Sistem, daha üst düzey modüllerin alt düzey modüllerle birleşimiyle tamamlanır.

- Saydam Kutu Testlerle Başlar, Kara Kutu Testlere Dönüşebilir: Entegrasyon testleri genellikle saydam kutu testlerle başlar ve ilerledikçe kara kutu testlere dönebilir.

- Yüksek Seviye Tasarımı Gerektirir: Bu yaklaşım daha üst düzey tasarım gerektirir ve genellikle üst düzey modüllerin nasıl çalıştığına odaklanır.

Aşağıdan yukarı yaklaşımın avantajlarına aşağıda yer verilmektedir:

- Erken Test Edilebilirlik: Alt düzey modüllerin birim testleri erken aşamalarda başladığı için hatalar daha erken tespit edilebilir ve düzeltilebilir.

- Modül Odaklı Yaklaşım: Modül odaklı bir yapıya sahip olduğundan, her bir modülün bağımsız olarak test edilmesi ve geliştirilmesi daha kolaydır.

- Daha İyi Yüzdelik Tamamlama İzleme: Hangi modüllerin tamamlandığını ve test edildiğini izlemek daha kolaydır.

- Kara Kutu Testlere Başlama Kolaylığı: Entegrasyon testleri kara kutu testlerle başlar ve daha sonra saydam kutu testlere dönüşebilir, bu da testlerin daha hızlı başlamasını sağlar.

Aşağıdan yukarı yaklaşımın dezavantajları:

- Entegrasyon Zorluğu: Üst düzey modüllerin bir araya gelmesi ve entegrasyonu daha geç bir aşamada gerçekleştiği için bu süreç zorlu olabilir.

- Üst Düzey Tasarım Gerektirir: Alt düzey modüllerin başarılı bir şekilde tasarlanması ve test edilmesi için daha önce üst düzey tasarımın tamamlanmış olması gerekir.

Yukarıdan Aşağı Yaklaşımın Avantajları:

- Sistem Odaklı Yaklaşım: İlk olarak sistemin en üst düzey işlevselliği test edildiği için sistemin daha yüksek seviyeli işlevselliği hızla doğrulanabilir.

- Üst Düzey Tasarım Gerektirmez: Daha önce alt düzey tasarımın tamamlanmasına gerek yoktur, bu nedenle üst düzey tasarım daha erken aşamalarda oluşturulabilir.

- Entegrasyon Kolaylığı: Üst düzey modüllerin entegrasyonu daha erken başladığı için entegrasyon süreci daha düşük seviyelerde gerçekleşir ve daha az karmaşıktır.

- Kara Kutu Testlere Başlama Kolaylığı: Entegrasyon testleri saydam kutu testlerle başlar ve ilerledikçe kara kutu testlere dönebilir, bu da testlerin daha hızlı başlamasını sağlar.

Yukarıdan Aşağı Yaklaşımın Dezavantajları:

- Erken Hata Tespit Zorluğu: Üst düzey tasarımın tamamlanmasını beklemek, alt düzey hataların daha sonra tespit edilmesine neden olabilir.

- Modül Odaklı Değil: Bu yaklaşım alt düzey modüllerin bağımsız olarak geliştirilmesini ve test edilmesini engelleyebilir.

Hangi yaklaşımın kullanılacağı, projenin gereksinimlerine, takvimine ve karmaşıklığına bağlı olarak değişebilir. Bu nedenle, her iki yaklaşımın avantajları ve dezavantajları dikkate alınarak doğru seçim yapılmalıdır.

Bilgi sistemleri denetçisinin bu testlerde dikkat etmesi gereken hususları:

- Hata raporlarını kayıt ve takip için kullanılan prosedürleri gözden geçirmek,

- Sistem güvenliğinin tasarlandığı gibi çalıştığını doğrulamak,

- Doğruluğu için döngüsel işleme ve kritik raporları doğrulamak,

- Paralel test sonuçlarını ve kullanıcı kabul testini gözden geçirmek,

- Sistemin son kullanıcıları ile yeni yöntemleri, prosedürleri ve kullanım talimatlarını anladıklarını belirlemek için görüşmek,

Test planını, hata raporlarını, son kullanıcı belgelerini ve eksiksizlik ve doğruluk için kullanılan prosedürleri gözden geçirmek,

- Dahili kontroller için testlerin planlanıp gerçekleştirildiğini belirlemek için birim ve sistem test planlarını gözden geçirmek,

- Kullanıcı kabul testini incelemek ve kabul edilen yazılımın uygulama ekibine teslim edildiğinden emin olmak (Satıcı bu sürümü değiştirememelidir.),

- Kontrol toplamlarını ve dönüştürülmüş veriyi toplamak.

2.1.8. Üretim Ortamına Aktarım

Karmaşık BT sistemlerinin etkin ve verimli şekilde geliştirilmesi ve bakımı, bir işletme içinde titiz yapılandırması, değişiklik ve sürüm yönetimi süreçlerinin uygulanması ve uyum sağlanmasını gerektirir. Bu süreçler, sistemi oluşturan BT bileşenlerinin nitelikleri (donanım, yazılım, bellenim ve fiziksel bağlantı ortamı kablosu, fiber, radyo frekansı [RF] dahil olmak üzere ağ bağlantısı) üzerinde sistematik, tutarlı ve açık bir kontrol sağlar. Bilgi işlem ortamlarının yapılandırma durumu bilgisi, bu sistemlerin zamanında bakımının sağlanmasının yanı sıra sistem güvenilirliği, kullanılabilirliği ve güvenliği için kritik öneme sahiptir. BT sistemlerindeki değişiklikler, iş süreçlerinde istenmeyen sonuçları en aza indirmek için dikkatlice değerlendirilmeli, planlanmalı, test edilmeli, onaylanmalı, belgelenmeli ve iletilmelidir.

Bilgi sistemleri denetçisi, yapılandırma, değişiklik ve sürüm yönetimini yönetmek için mevcut araçlardan ve geliştirme personeli ile üretim ortamı arasında SoD'yi sağlamak için mevcut kontrollerden haberdar olmalıdır.

Kontrol etme süreci; donanım, ağ ve sistem mimarlarının hem donanım varlığı hem de envanter izleme sistemlerinde yapılan değişiklikleri veya güncellemeleri gözden geçirmesi ve onaylamasıyla paralel olarak kod düzenlemelerini önler veya yönetir.

Check-in (giriş), bir öğeyi kontrollü ortama taşıma işlemidir. Bir değişiklik gerektiğinde (ve bir değişiklik kontrol formu tarafından desteklendiğinde), konfigürasyon yöneticisi öğeyi kontrol eder Değişiklik yapıldıktan sonra farklı bir sürüm numarası aracılığıyla kontrol edilebilir. Teslim alma işlemi

aynı zamanda eş zamanlı kod düzenlemelerini de önler veya yönetir. Donanım ile ağ ve sistem mimarları hem donanım varlığı hem de envanter takip sistemlerinde yapılan değişiklikleri veya güncellemeleri inceler ve onaylar.

Yapılandırma yönetiminin çalışması için yönetim desteği çok önemlidir. Konfigürasyon yönetimi süreci, bir konfigürasyon yönetim planı ve işletim prosedürleri geliştirip takip ederek uygulanır. Bu plan sadece geliştirilen yazılımla sınırlı kalmamalı, tüm sistem dökümantasyonunu, test planlarını ve prosedürlerini de içermelidir.

Özetle iş surecilerinde istenmeyen sonuçların en aza indirilmesi için BT sistemlerindeki değişiklikler dikkatlice değerlendirilmeli, planlanmalı, test edilmeli, onaylanmalı, belgelenmeli ve anlatılmalıdır. Bilgi sistemleri denetçisinin, geliştirme personeli ile üretim ortamı arasında görevler ayrılığını (SoD) sağlamak için konfigürasyon, değişim ve sürüm yönetimi ve mevcut kontrollerin yönetimi için mevcut araçların farkında olması gerekir.

Veri Taşıma:

Veri dönüştürmenin amacı, verilerin anlam ve bütünlüğü korunurken, sistemdeki mevcut verilerin yeni formata, kodlamaya ve yapıya dönüştürülmesidir. Veri dönüştürme işlemi, dönüştürülen verilerin doğruluğunun ve eksiksizliğinin doğrulanmasına izin veren denetim izleri ve günlükler gibi bazı araçlar sağlamalıdır. Bu doğruluk ve eksiksizliğin doğrulanması, manuel süreçlerin, sistem yardımcı programların, satış araçlarının ve tek seferlik özel uygulamaların kombinasyonuyla gerçekleştirilebilir.

Veri taşıma projesi dikkatli bir şekilde planlanmalı ve aşağıdaki riskleri en aza indirmek için uygun metodolojiler ve araçlar kullanmalıdır:

- Eski ve taşınmış işlemler arasında yaşanan çatışmalar ve anlaşmazlıklar,

- Verilerin tasıma sürecinde yaşanan veri tutarsızlığı ve bütünlüğü kaybı,

- Rutin işlemlerin kesintisi,

- Verinin güvenlik ve gizlilik ihlali.

Uygulamaya Almanın Planlanması:

Üretim ortamını kurma adımlarının her biri, kimin sorumlu olacağı, adımın nasıl doğrulanacağı ve geri dönüş prosedürü dahil olmak üzere belgelendirilmelidir. Gerçekleştirme tarihinden çok önce bir gerçekleştirme planı hazırlanmalıdır. Başarılı testten sonra, sistem işletmenin değişim kontrol prosedürlerine göre gerçekleştirilir. Bir geri dönüş senaryosu gerçekleştirilmelidir.

Uygulamaya alma sürecinin başarılı bir şekilde tamamlanabilmesi aşağıdaki hususlara özellikle dikkat edilmelidir:

- Testlerin Tamamlanması: Üretim ortamına almadan önce, yazılımın ve sistemlerin tüm testlerinin başarılı bir şekilde tamamlandığından emin olunmalıdır. Bu, yazılımın işlevselliğinin ve güvenilirliğinin sağlandığını gösterir.

- Eğitim ve İşletme Hazırlıkları: Sistem uygulamaya alınmadan önce, son kullanıcıların eğitilmesi ve işletme ekibinin gerekli hazırlıkları yapması önemlidir. Kullanıcılar, sistemi etkin bir şekilde kullanabilmeli ve potansiyel sorunları bildirebilmelidir.

- Yedekleme ve Geri Dönüş Planı: Uygulamaya alma sırasında herhangi bir aksilik durumunda sistemin eski haline dönebilmesi için yedekleme ve geri dönüş planları hazırlanmalıdır. Bu planlar, veri kaybını önler ve iş sürekliliğini sağlar.

- Değişim Kontrol Prosedürleri: Uygulamaya alma süreci, değişiklik kontrol prosedürlerine uygun olarak yönetilmelidir. Bu prosedürler, yapılan değişikliklerin izlenmesini, onaylanmasını ve dokümante edilmesini sağlar.

- İzleme ve Geri Bildirim: Uygulamaya alma sonrasında sistem sürekli olarak izlenmeli ve kullanıcıların geri bildirimleri dikkate alınmalıdır. Bu, olası sorunların hızlı bir şekilde tespit edilmesine ve çözülmesine olanak tanır.

- Sorun Giderme ve Destek: Uygulamaya alma sonrası dönemde kullanıcıların yaşadığı sorunların hızlıca giderilebilmesi için bir destek mekanizması oluşturulmalıdır. Kullanıcılar, sistemle ilgili herhangi bir sorunla karşılaştıklarında bu destek mekanizmasını kullanabilirler.

Geçiş (Canlıya Geçiş veya Sistem Geçişi) Teknikleri:

a. Paralel Geçiş

Paralel geçiş tekniği, eski sistemin çalıştırılması, ardından hem eski hem de yeni sistemlerin eş zamanlı çalıştırılmasıdır. Bu yaklaşım 2 sistemin karşılaştırılmasına olanak tanır. Yeni sistemin çalıştığına dair güven kazanıldıktan sonra tamamen yeni sisteme geçilir Yeni sistemin istenildiği gibi çalışmaması durumunda eski sisteme dönüşe müsaade ettiği için riski yüksek değildir. İki sistemin aynı anda çalışmasının maliyeti yüksektir ancak risk daha düşüktür.

Örnek: Bir büyük perakende zinciri, mevcut mağaza yönetim sistemini güncellemeye karar verir. Paralel geçiş tekniğini kullanarak, eski mağaza yönetim sistemi aynı anda yeni sistemle çalışır. Her bir mağaza aynı anda hem eski hem de yeni sistemi kullanır. Bu sayede mağazalar yeni sistemi kullanmayı öğrenirken iş sürekliliği sağlanır. Güven kazanıldığında, tamamen yeni sisteme geçiş yapılır ve eski sistem kademeli olarak kapatılır.

b. Fazlı (Aşamalı) Geçiş

Fazlı (aşamalı) geçiş tekniği, alt sistemlerin sırayla yeni sisteme geçişidir. Eski sistemden yeni sisteme sırayla modül bazında geçiş sağlanır. Son modüle ulaşana kadar aşamalı olarak kaldırılır.

Örnek: Bir üniversite, öğrenci bilgi sistemi üzerinde bir güncelleme yapmayı planlar. Fazlı geçiş tekniğini kullanarak, önce kayıt ve ders programı modüllerini yeni sisteme geçirir. Bu modüller yeni sistemin bir parçası olarak çalışırken, diğer modüller eski sistemde kalmaya devam eder. Sonraki aşamada, öğrenci notları ve transkript modülleri yeni sisteme geçirilir. Bu şekilde, her aşama sırasında sistemin bir bölümü güncellenir, böylece tüm sistemi aynı anda değiştirmek yerine risk azaltılır.

c. Ani (Doğrudan) Geçiş

Ani geçiş tekniğinde, yeni sistem, bir son tarih ve saatte eski sistemden değiştirilir ve yeni sisteme geçiş gerçekleştiğinde eski sistem durdurulur. Genel olarak eski sistem görevini yerine getiremediğinde tercih edilir. Ani geçiş yaklaşımının riski yüksektir. Eğer sistem istenilen verimi elde edemezse geri dönüş maliyeti çok yüksek olacaktır.

Örnek: Bir telekomünikasyon şirketi, yeni bir müşteri hizmetleri platformuna geçmeyi planlar. Ani geçiş tekniğini kullanarak, belirli bir tarihte ve saatte eski platformu tamamen kapatır ve yeni platformu başlatır. Bu geçiş anında ve ani bir şekilde gerçekleşir. Özellikle eski platformun işlevselliği bozulmuşsa veya güncellemesi zorunlu hale gelmişse tercih edilir. Ancak bu yöntem yüksek risk içerir, çünkü geçiş sırasında herhangi bir sorun ciddi kesintilere neden olabilir.

d. Parça Parça Geçiş:

Bu yöntemde, sistemdeki farklı bileşenler veya modüller ayrı ayrı yeni sisteme geçirilir. Tüm sistemin aynı anda geçişini beklemek yerine, belirli bileşenler veya modüller zamanla yeni sisteme entegre edilir. Yani alt sistemlerden biri yeni sisteme geçer ve diğer alt sistemler ise bir süre daha eski sistemde devam eder. Bu yöntem, büyük ölçekli sistemler söz konusu olduğunda tercih edilir. Bu yöntemin dezavantajı ise geçiş süresinin uzamasıdır.

Örnek Durum: Bir büyük e-ticaret platformu, ödeme işlemleri modülünü yeni bir sistemle değiştirmeye karar verir. İlk olarak, yalnızca ödeme modülü yeni sisteme geçirilir ve diğer modüllerle entegre edilir. Daha sonra diğer modüller parça parça yeni sisteme entegre edilir.

e. Gerçek Zamanlı Geçiş:

Bu yöntemde, eski sistemden yeni sisteme geçiş süreci, kullanıcıların mevcut işlemlerine kesinti veya gecikme olmadan devam etmesine olanak tanır. Yani geçiş işlemi arka planda gerçekleştirilir ve kullanıcılar bu geçişi fark etmez.

Örnek Durum: Bir banka, yeni bir çevrimiçi bankacılık platformuna geçmeye karar verir. Kullanıcılar bankacılık işlemlerine herhangi bir kesinti yaşamadan devam edebilirler, ancak banka sistemleri arka planda yeni sisteme geçer.

Bilgi sistemleri denetçisi, uygulamaya alınmadan önce uygun onayların alındığını doğrulamalı ve aşağıdakileri yapmalıdır:

- Sistemi üretime almadan önce doğru ve tam olduklarından emin olmak için tüm veri dönüşümünü doğrulamak,

- Tamlığını sağlamak ve test aşamasındaki tüm güncellemelerin dahil edildiğinden emin olmak için tüm sistem belgelerini incelemek,

- Üretim zamanlamasının yürütülmesinde kullanılan sistem parametrelerinin yanı sıra, sistemi zamanlamak ve çalıştırmak için kullanılan programlanmış prosedürleri gözden geçirmek.

Devreye Alma Planı:

Sürümün devreye alınması için aşağıdaki aşamalar gerçekleştirilir:

Rollerin Belirlenmesi: Devreye alma sürecinde, hangi ekip üyelerinin hangi rolleri üstleneceği belirlenmelidir. Bu rollerin belirlenmesi, her bir kişinin ne yapması gerektiğinin açıkça tanımlanması anlamına gelir. Özellikle proje yönetimi, teknik uzmanlar, test ekipleri gibi farklı rollerin net bir şekilde tanımlanması gerekmektedir. Bu, işlerin koordinasyonunu ve sorumlulukların net bir şekilde belirlenmesini sağlar.

Gerekli Yetenek/Becerilerin Kazanılması ve Eğitimlerin Verilmesi: Devreye alma sürecinin başarılı olabilmesi için ekip üyelerinin gerekli yeteneklere ve becerilere sahip olmaları önemlidir. Eğer eksik yetenekler varsa, bu eksiklikleri gidermek için ekip üyelerine eğitimler verilmelidir. Özellikle yeni teknolojilerin veya sistemlerin kullanılması gerekiyorsa, bu alandaki eğitimler ayrı bir öneme sahiptir. Gerekli yeteneklere sahip ekip üyeleri, sürecin sorunsuz bir şekilde ilerlemesine katkı sağlar.

İş Yükünün Rol ve Sorumluluklara Göre Dağıtılması: Devreye alma sürecinde işlerin adil bir şekilde dağıtılması önemlidir. Her bir rol sahibi, ne yapması gerektiğini ve hangi sorumlulukları taşıdığını net bir şekilde bilmelidir. İş yükünün dengeli bir şekilde dağıtılması, ekip üyelerinin verimliliğini artırır ve proje sürecinin etkin bir şekilde yönetilmesini sağlar. Ayrıca, iş yükünün dağıtımı sırasında öncelikler belirlenmeli ve acil görevler önceliklendirilmelidir.

Fazlı Bir Geçiş Planı Oluşturulması: Devreye alma süreci boyunca herhangi bir sorunun erken tespit edilmesi ve çözülmesi kritik öneme sahiptir. Bu nedenle, fazlı bir geçiş planı oluşturulmalıdır. Bu plan, projenin farklı aşamalarının belirli bir sırayla ve kontrollü bir şekilde gerçekleştirilmesini sağlar. Aynı zamanda, olası riskleri azaltmaya yardımcı olur. Fazlı geçiş planı, projenin hangi aşamalarında hangi testlerin yapılacağından, kullanıcılara ne zaman eğitim verileceğine kadar detaylı bilgiler içermelidir. Bu plan, projenin başarılı bir şekilde sonuçlanmasına katkı sağlar ve kullanıcılar veya paydaşlar için minimum kesinti sağlar.

Eğitim Süreci:

Projenin getirdiği yeni sistemin veya projenin mevcut sisteme getirdiği yenilikler kapsamında eğitim ihtiyaçları doğabilmektedir. Bunun yanı sıra rol ve sorumlulukların da bu kapsamda gözden geçirilmesi gerekmektedir. Bu kapsamda, gereksinimler dikkate alınarak eğitim planı hazırlanmalıdır. Eğitim planı hazırlanırken aşağıda yer alan tüm detayların bulunması beklenmektedir:

İçerik: Eğitim planının temel taşlarından biri, eğitimin içeriğidir. Hangi konuların ele alınacağı, eğitilecek personelin ihtiyaçlarına ve projenin gereksinimlerine dayalı olarak belirlenmelidir. İçerik, yeni sistem veya süreçle ilgili gereksinimler, kullanım talimatları, en iyi uygulamalar ve diğer önemli bilgileri içerebilir.

Zaman Planı Bilgisi: Eğitim planlaması yapılırken, eğitimlerin ne zaman yapılacağına dair bir zaman çizelgesi belirlenmelidir. Bu zaman çizelgesi, projenin genel zaman çizelgesiyle uyumlu olmalıdır ve eğitimlerin hangi aşamalarda yapılacağını net bir şekilde göstermelidir.

Süre: Her bir eğitim oturumunun süresi belirlenmelidir. Eğitimlerin uzunluğu, içerik karmaşıklığına, katılımcıların tecrübesine ve diğer faktörlere bağlı olarak değişebilir. Süre, katılımcıların bilgiyi anlaması ve içselleştirmesi için yeterli olmalıdır.

Teslim Şekli: Eğitimlerin nasıl teslim edileceği belirtilmelidir. Eğitimler yüz yüze mi yapılacak, çevrimiçi mi olacak yoksa bir kombinasyon mu tercih edilecek? Teslim yöntemi, katılımcıların erişimine ve gereksinimlerine uygun olmalıdır.

Eğiticiyi Eğitme Kavramı: Eğitimcilerin, eğitim materyallerini ve konularını nasıl ileteceği önemlidir. Eğitimciler, eğitim materyallerini ve yöntemlerini etkili bir şekilde kullanabilmek için eğitilmelidirler. Bu, eğitim kalitesini artırabilir.

Geri Bildirim Mekanizması: Eğitim sürecinin başarısını ölçmek ve iyileştirmek için bir geri bildirim mekanizması oluşturulmalıdır. Katılımcıların eğitim sonrası deneyimlerini değerlendirmek ve eksiklikleri tespit etmek için geri bildirim formları veya anketler kullanılabilir. Bu geri bildirimler, eğitim materyallerinin ve yöntemlerinin sürekli olarak iyileştirilmesine yardımcı olur.

Uygulama ve Simülasyon: Eğitim süreci, katılımcılara teorik bilgi vermenin ötesine geçmelidir. Uygulamalar, simülasyonlar veya canlı senaryolar kullanılarak katılımcıların öğrendiklerini pratikte uygulamalarına olanak tanınmalıdır. Bu, öğrenilen bilginin daha iyi içselleştirilmesini sağlar.

Eğitim Materyalleri ve Kaynakları: Eğitim süreci için gerekli materyaller ve kaynaklar sağlanmalıdır. Bu, eğitim sunumları, dokümantasyon, videolar, örnek senaryolar ve yardımcı kaynakları içerebilir. Eğitim materyalleri, katılımcıların öğrenme sürecini desteklemek için kolayca erişilebilir olmalıdır.

Çevrimiçi Eğitim Seçenekleri: Eğitim süreci, katılımcılara çevrimiçi eğitim seçenekleri sunarak daha esnek hale getirilebilir. Bu, uzaktan çalışan veya farklı coğrafyalarda bulunan katılımcıların eğitimlere daha kolay katılmasını sağlar.

Sürekli Destek ve Güncelleme: Eğitim süreci, projenin veya sistemlerin yaşam döngüsü boyunca sürekli destek ve güncelleme gerektirir. Bu, yeni özelliklerin eklenmesi veya sistemde değişiklikler yapılması durumunda güncel eğitimlerin sağlanmasını içerir.

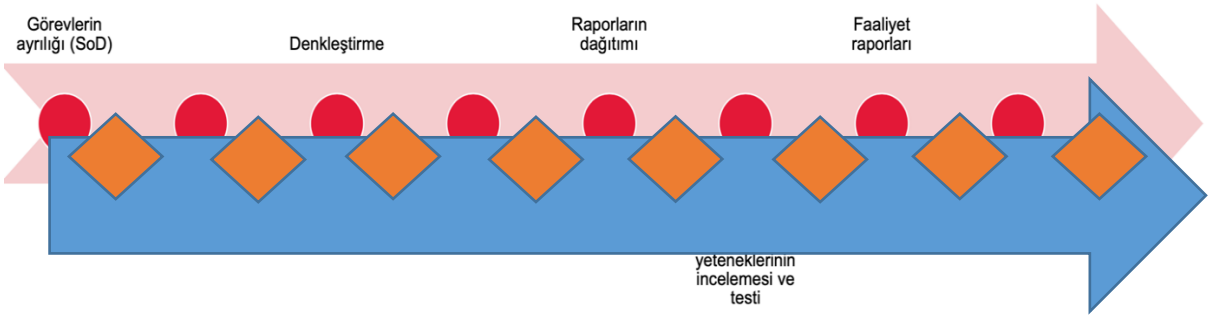
Bu plan yeni yapı için rol tanımlarını beceri profillerini ve boşluk analizi sonuçlarını dikkate almalıdır. Plan hazırlanırken eğitilmesi gereken personelin mevcut sistemi çalıştırması gerektiği de dikkate alınarak iş gücü kaybına neden olmayacak şekilde bir koordinasyon yürütülmelidir.

Geliştirme sürecinde yer alan tüm birimler, her zaman son kullanıcıdan birkaç adım önde olmalıdır. Bir sonraki adımda; uygulamanın çoklu ortamlara, internet ortamına veya başka etkileşimli ortamlara taşınması istenebilir. Böylece, belki başlangıçta düşük maliyetlerde hesaplanan uygulama, değerini de arttıracak teknik özelliklere sahip olabilir.

Son Kullanıcı Eğitimi

Bu eğitimin amacı son kullanıcının yetkinliği artırılarak sistemi daha etkin ve verimli kullanabilmesini sağlamaktır. Bu sürecin başarılı olabilmesi için geliştirme sürecinden başlanarak bir eğitim planı hazırlanmalıdır. Bu şekilde geliştirilecek bir eğitim stratejisinde zamanlama, kapsam ve dağıtım mekanizmaları önem arz etmektedir.

Eğitim için öncelikle bir pilot uygulama yapılabilir daha sonra pilot gruptan gelen geri bildirimler ışığında eğitim üzerinde düzenlemeler yapılıp ve eğitim planında da ayarlamalara gidilebilir. Eğitim verme de zamanlama çok önemlidir. Eğitim erken verilirse kullanıcılar sistem üretime geçene kadar çoğunu unuturlar. Eğitim geç verilirse pilot grubun geri bildirimleri ile eğitimi olgunlaştıracak zaman kalmaz ve eğitimde istenilen başarı elde edilemez. Ayrıca eğitimler kullanıcıların organizasyon içindeki rollerini, beceri düzeylerini ve ihtiyaçlarını karşılayacak şekilde düzenlenmelidir.



Kullanıcı Prosedürleri ve Dökümantasyon Şeması

Sistemin son kullanıcı ve sistem yöneticileri için hazırlanan kılavuzları kapsamaktadır. Geliştirme hakkında bilgi verir, gereksinimleri, teknik özellikleri, iş gerekçelerini ve kılavuzları kullanıcı belgelerini, sorun giderme kılavuzlarını, kurulum ve başvuru kılavuzlarını içerir.

Bilgi sistemleri denetçisi tarafından gözlemlenmesi ve test edilmesi gereken kullanıcı prosedürleri ve dökümantasyonlar aşağıda verilmektedir.

• **SoD** - Hiç kimsenin aşağıda belirtilen süreçlerden birden fazlasını gerçekleştirmeye yetkili olmamasını sağlar:

- Başlatma (Initiation): Başlatma süreci, bir iş veya işlemi başlatma yetkisine sahip olan kişilerin kimler olduğunu ve bu yetkinin nasıl verildiğini içerir. Bir bilgi sistemleri denetçisi, bu süreci inceleyerek kimlerin iş veya işlem başlatma yetkisine sahip olduğunu belirlemeli ve bu yetkilerin nasıl verildiğini, izlendiğini ve gerektiğinde iptal edildiğini doğrulamalıdır.

- Yetkilendirme (Authorization): Yetkilendirme süreci, bir iş veya işlemi gerçekleştirmek için gerekli olan yetkilendirmeleri içerir. Bu, kimlerin belirli görevleri veya işlemleri yapma yetkisine sahip olduğunu belirlemeyi ve bu yetkilerin doğru şekilde belgelendirildiğini ve uygulandığını doğrulamayı içerir.

- Doğrulama/dağıtım (Verification/Distribution): Doğrulama ve dağıtım süreçleri, belirli bilgilerin veya kaynakların doğru bir şekilde doğrulandığını ve gerektiğinde doğru kişilere dağıtıldığını sağlar. Bir bilgi sistemleri denetçisi, bu süreçleri inceleyerek bilgilerin nasıl doğrulandığını, kimlerin bu doğrulamayı yapabileceğini ve bilgilerin gerektiğinde güvenli bir şekilde dağıtıldığını doğrulamalıdır.

İş tanımlarının gözden geçirilmesi, yetki seviyeleri ve prosedürlerin gözden geçirilmesi, SoD'nin varlığı ve uygulanması hakkında bilgi sağlayabilir.

• **Giriş yetkisi** - Girilen belgelerde tanımlı yetkilendirme veya benzersiz parolalar kullanılması gerekir. Uygun yetkilendirme girdi belgelerinin bir örneğine bakılarak veya bilgisayar erişim kuralları gözden geçirilebilir. Veri doğrulamaları takip edilmeli ve geçersiz işlemler incelenerek otomatik günlüğe kaydetmesi sağlanmalıdır. Bu geçersiz işlem faaliyet raporu, yönetimsel incelemenin kanıtı için test edilmelidir. Aşırı geçersiz işlemler, verimliliği artırmak için doğrulama ve düzenleme rutinlerinde değişiklik yapılması gerektiğini gösterebilir.

• **Denkleştirme** – Çalıştırılan kontrol toplamları ile ve diğer uygulama toplamlarının zamanında uygunluğu doğrulanmalıdır. Bu, bağımsız denkleştirmelerle veya geçmiş uzlaşmaları gözden geçirilerek test edilebilir.

• **Hata denetimi ve düzeltme** - Düzeltme ve yeniden gönderme şeklinde uygun inceleme kanıtlarını içeren rapor üretilmelidir. Giriş hataları ve "*red*"ler yeniden gönderilmeden önce gözden geçirilmelidir. Yapılan düzeltmelerin yönetim tarafından incelenmesi ve yetkilendirilmesine ilişkin kanıtlar dökümante edilmelidir. Bu eforun testi, geçmiş hata düzeltmelerini yeniden düzenleyerek veya gözden geçirerek gerçekleştirilebilir.

• **Raporların dağıtımı** - Kritik çıktı raporları güvenli bir alanda üretilerek saklanır ve yetkili bir şekilde dağıtılır. Dağıtım süreci, dağıtım çıktı günlükleri gözlemlenerek ve gözden geçirilerek test

edilebilir. Çevrimiçi çıktı raporlarına erişim kısıtlanmalıdır. Çevrimiçi erişim kuralları incelenerek veya kullanıcı çıktısı izlenerek test edilebilir.

• **Erişim yetkileri ve yeteneklerinin incelenmesi ve testi** - Erişim seviyeleri hakkında bilgi sağlar (kontrol tabloları). Erişim, iş tanımlarına dayalı olmalı ve SoD'yi sağlamalıdır. Test erişiminin yönetimin istediği gibi verildiğinden emin olmak için kuralların gözden geçirilmesi yolu ile gerçekleştirilebilir.

• **Faaliyet raporları** - Kullanıcı bazında etkinlik hacmi ve çalışma saatleri hakkında ayrıntılar sağlamaktadır. Faaliyetlerin yalnızca yetkilendirilmiş saatlerde yapıldığına güvence vermek için düzenli olarak gözden geçirilmelidir.

• **İhlal raporları** - Başarısız ve yetkisiz erişim girişimlerini terminalin konumu erişim girişim tarihi ve saati gibi detayları içerir şekilde gösterir. Bu raporlar yönetimin gözden geçirmesine yönelik kanıt dökümanı olarak değerlendirilebilir. Tekrarlanan yetkisiz erişim ihlalleri, erişim kontrollerini atlama girişimlerini gösterebilir.

2.1.9. Kritik Başarı Faktörleri

Kritik başarı faktörleri, iş verimliliği ekonomik değer, müşteri hizmetleri ve kaliteyi içermektedir. Gereksinimler için, yöntemlerin aktarılabileceği bir şablon oluşturması SDLC'nin sağladığı temel avantajıdır. SDLC'nin kritik başarı faktörleri aşağıdakileri içermelidir:

Proje Başlangıcında İş İçin Gerekli Belgelerin Temini: SDLC'nin başlangıcında, proje için gerekli belgelerin toplanması ve hazır olması, projenin başarısını etkileyen kritik bir faktördür. Bu belgeler, projenin gereksinimlerini, hedeflerini ve kapsamını belirlemeye yardımcı olur. Eksik veya yanlış belgeler, projenin yanlış yönlendirilmesine neden olabilir.

Karar Vericilerin Katılımı: Karar vericilerin, projenin farklı aşamalarında aktif olarak yer aldığı ve gerekli kararları aldığı bir süreç, projenin etkili bir şekilde yönetilmesini sağlar. Bu, proje ekibinin doğru yönlendirilmesini ve gerektiğinde kararların hızla alınmasını sağlar. Karar vericilerin katılımı, proje risklerini azaltabilir ve projenin zamanında ve bütçe dahilinde tamamlanmasına yardımcı olabilir.

Yetkili Personelin Katılımı: Proje yaşam döngüsü boyunca, ilgili ve yetkili personelin proje süreçlerine ve komitelere katılması kritik bir faktördür. Bu, projenin gereksinimlerini daha iyi anlamamıza ve daha etkili bir şekilde yönetmemize yardımcı olur. Ayrıca, yetkili personelin katılımı, gecikmelerin önlenmesine ve projenin başarısına katkı sağlar.

Kritik Başarı Faktörleri Ölçümleri:

SDLC kapsamında kritik başarı faktörleri için aşağıdaki hususlarda yapılacak ölçümler takip edilebilecektir.

a. Üretkenlik

- Her Kullanıcı İçin Harcanan Para: Proje için harcanan bütçenin kullanıcı sayısına bölünmesi, her bir kullanıcının projeye ne kadar maliyet getirdiğini gösterir.

- Kullanıcı Bazında Yapılan İşlemler İçin Hareket Sayısı: Kullanıcıların sistemi ne kadar yoğun kullandığını ve hangi işlemleri gerçekleştirdiğini gösterir.

- Aylık Toplam İşlem Sayısı: Sistem üzerinden gerçekleştirilen toplam işlem sayısı, sistemin ne kadar aktif olduğunu ve kullanıldığını yansıtır.

Bu ölçümler, projenin maliyet-etkinliğini ve kaynakların nasıl kullanıldığını değerlendirmeye yardımcı olur.

b. Kalite

- Anlaşma Sağlanamayan Konuların Sayısı: Proje sürecindeki anlaşmazlıkların ve çatışmaların sayısı, iletişim ve işbirliği eksikliklerini gösterebilir.

- Kötüye Kullanım Ya Da Sahtecilik Kullanım Sayısı: Sistem üzerinden tespit edilen kötüye kullanım veya sahtecilik girişimlerinin sayısı, güvenlik açıkları ve risklerin bir göstergesidir.

- Tutarsızlık Sayısı: Sistemdeki tutarsızlıkların sayısı, kalite kontrolündeki eksiklikleri gösterebilir.

Bu ölçümler, projenin kalitesini ve güvenliğini değerlendirmeye yardımcı olur.

c. Ekonomik Değer

Yönetim Maliyetleri: Projenin yönetim ve idari maliyetleri, ekonomik değeri etkileyen önemli bir faktördür. Bu ölçüm, projenin ekonomik değerini ve maliyet-etkinliğini değerlendirmeye yardımcı olur.

d. Müşteri Hizmetleri

- Müşteri Problemleri İle İlgili Müşteriye Geri Dönüş Süresi: Müşteri problemlerine ne kadar hızlı yanıt verildiği ve çözüldüğü, müşteri hizmetleri kalitesini gösterir.

- Kullanıcılar İle İletişim Periyodu: Kullanıcılarla ne sıklıkla iletişim kurulduğu, müşteri memnuniyetini ve kullanıcı deneyimini etkiler.

Bu ölçümler, müşteri hizmetlerinin ve kullanıcı memnuniyetinin izlenmesine yardımcı olur.

2.1.10. Sistem Geliştirme Süreçlerindeki Riskler

Potansiyel Riskler:

Sistem geliştirme süreçleri için süreci tasarlarken ve geliştirirken birçok potansiyel riskle karşılaşılabilir. Bu riskler aşağıdaki gibi özetlenebilir:

a. İş Riski:

Yeni sistem, kullanıcı iş ihtiyaçlarını, gereksinimlerini ve beklentilerini karşılayamama riski taşır. Kullanıcıların ihtiyaçları doğru anlaşılmaz veya eksik karşılanırsa, yeni sistem işletme verimliliğini düşürebilir.

Örnek 1: Bir eğitim kurumu, mevcut öğrenci bilgi yönetim sisteminin yerine yeni bir yazılım uygulamayı planlar. Ancak, yeni yazılımın öğrenci kayıt süreçlerini eksik veya hatalı bir şekilde ele alması nedeniyle, öğrencilerin kayıt işlemleri aksamış ve okulun işleyişi olumsuz etkilenmiştir.

Örnek 2: Bir otomobil üreticisi, üretim hattına otomatik bir montaj robotu eklemeyi düşünüyor. Ancak, yeni robotun montaj işlemi sırasında hata yapma olasılığı yüksek olduğundan, bu süreçte hatalı montajlar ve ürün kayıpları riski taşımaktadır.

b. Proje Riski:

Projenin başarıyla bitirilmesini engelleyici faktörler proje riskini oluşturur. Proje faaliyetleri, başlangıçta belirlenen tasarım ve geliştirmelerin dışına çıkabilir. Bu, proje süresinin uzamasına, maliyet artışlarına ve proje hedeflerinin gerçekleştirilmemesine neden olabilir.

Örnek 1: Bir yazılım geliştirme projesi, başlangıçta belirlenen bütçenin ve zaman çerçevesinin üzerine çıkarak planlanan teslim tarihini kaçırır. Bu nedenle, proje bitmeden önce ek kaynaklara ve süre uzatmalarına ihtiyaç duyulur.

Örnek 2: Bir inşaat projesi, beklenmedik hava koşulları nedeniyle planlandığı gibi ilerleyemez. Bu gecikmeler, proje bütçesinin artmasına ve iş bitim tarihine ulaşmanın zorlaşmasına neden olabilir.

c. Tedarikçi Riski:

Gereksinimlerin ve beklentilerin net olarak iletilmemesi veya tedarikçilerin geç teslimat yapması gibi nedenlerle tedarikçi ilişkileri risk taşıyabilir. Bu durum, proje sürecinin aksamalarına ve maliyet artışlarına yol açabilir.

Örnek 1: Bir şirket, yeni bir tedarikçiden özel üretilmiş bir makine satın alır. Ancak, tedarikçi, söz verilen teslimat tarihlerini sürekli olarak kaçırır ve bu nedenle üretim hattı aksamaya başlar.

Örnek 2: Bir restoran, taze deniz ürünleri tedarikçisi ile anlaşır ancak tedarikçi, belirli zamanlarda taze malzemeleri sağlama konusunda istikrarlı olamaz ve bu, restoranın menüsünün sınırlı olmasına neden olur.

d. Paydaşlar Riski:

Proje için gerekli olan girdilerin sağlanamaması, paydaşlar risk oluşturabilir. Paydaşların katılımı, projenin başarısı için kritik öneme sahiptir. Eğer gerekli bilgi, onay veya destek sağlanmazsa, projenin ilerlemesi zorlaşabilir.

Örnek 1: Bir yazılım projesi için kullanıcılar, proje başlangıcında gerekli kullanıcı verilerini ve beklentilerini iletmekte gecikirler. Bu, projenin gereksinimleri doğru bir şekilde karşılayamamasına neden olabilir.

Örnek 2: Bir şirket, bir yeni ürünün pazara sunulması için pazarlama ekibinin onayını beklerken pazarlama departmanı, gerekli geri bildirimleri zamanında sağlamaz ve bu nedenle ürün lansmanı gecikir.

e. Teknoloji Riski:

Yeni sistem, mevcut teknoloji altyapısına uyumsuz veya verimsiz olabilir. Bu durum, sistem performansını düşürebilir, güvenlik sorunlarına neden olabilir ve iş sürekliliğini etkileyebilir.

Örnek 1: Bir şirket, mevcut altyapısının iş gereksinimlerini karşılamak için yetersiz olduğunu fark eder ve yeni bir ERP (Enterprise Resource Planning) yazılımı uygulamaya karar verir. Ancak, yeni yazılım, mevcut donanımın kapasitesini aşar ve bu da sistem performansının düşmesine neden olur.

Örnek 2: Bir hava yolu şirketi, uçuşlarını yönetmek için yeni bir uçuş rezervasyon sistemi uygulamaya koyar. Ancak, yeni sistem, eski veritabanı sistemine uyumsuzdur ve bu da rezervasyon hatalarına ve uçuşların aksamalarına yol açar.

Bilgi sistemleri denetçisi tarafından takip edilmesi gereken konular aşağıdaki gibidir:

- Geliştirme ve Tasarım Adımlarının İncelenmesi ve Gözden Geçirilmesi: Bilgi sistemleri denetçisi, proje geliştirme süreçlerinin her aşamasını incelemeli ve tasarım adımlarının proje gereksinimlerini karşılayıp karşılamadığını değerlendirmelidir. Bu, projenin tasarımın belirlediği hedeflere uygun bir şekilde ilerlediğini ve tasarımın güvenlik, performans ve işlevsellik gereksinimlerini karşıladığını doğrulamak için önemlidir.

- Proje Aşamalarında Risk Analizlerinin Yapılması ve Kontrollerinin İncelenmesi: Bilgi sistemleri denetçisi, proje sürecinin farklı aşamalarında risk analizlerinin yapıldığını ve bu risklerin nasıl yönetildiğini incelemelidir. Ayrıca, risk analizlerine dayalı olarak alınan düzeltici önlemlerin ve kontrollerin etkin bir şekilde uygulandığını doğrulamalıdır. Bu, projenin beklenen risklere karşı güvende olduğunu ve gerektiğinde müdahale edildiğini gösterir.

- Proje Hedef ve Amaçlarının İncelenmesi: Bilgi sistemleri denetçisi, projenin hedeflerini ve amaçlarını anlamalı ve bu hedeflerin projenin her aşamasında nasıl izlendiğini ve değerlendirildiğini kontrol etmelidir. Proje hedeflerinin belirlenmesi ve takip edilmesi, projenin başarısını ölçmek ve hedeflere ulaşılıp ulaşılmadığını değerlendirmek için kritik bir unsurdur.

2.2. Sistem Bakım ve Destek Süreçleri

2.2.1. Sistem ve Uygulama Bakımı

Uygulamalardan sonra sistemler için sürekli geliştirme ya da bakım süreçleri başlatılır. Üretim, uygulama ve çalıştırılabilir kodun bütünlüğünü korurken, uygulama sistemlerinde değişiklik yönetimi süreci uygulanır. Proje tasarım aşamasında, değişikliklerin kayıt altına alınması ve gerçekleştirilmesi için standart bir değişiklik yönetimi süreci olmalıdır. Bu kapsamda bilgi sistemleri denetçisinin yapması gereken işlemler aşağıdaki gibidir:

- Mevcut Kontrollerin Gözden Geçirilmesi: Bilgi sistemleri denetçisi, sistemde bulunan güvenlik, bütünlük ve doğruluk kontrollerini düzenli olarak gözden geçirmelidir. Bu kontrollerin, sistem

tasarımına uygun bir şekilde çalıştığını doğrulamak, sistemin güvenli ve istikrarlı bir şekilde çalışmasını sağlar.

- Maliyet Fayda Analizlerinin Kontrol Edilmesi: Bilgi sistemleri denetçisi, sistemde yapılan değişikliklerin maliyetlerini ve faydalarını analiz etmelidir. Bu analizler, yönetim tarafından onaylanan projelerin maliyet etkinliğini değerlendirmeye yardımcı olur.

- Giriş ve Çıkış Kontrol Raporlarının Gözden Geçirilmesi: Sistem giriş ve çıkış kontrolleri, verilerin doğru bir şekilde işlendiğini ve işletmeye uygun olduğunu sağlar. Bilgi sistemleri denetçisi, bu raporları inceleyerek sistemin veri bütünlüğünü ve doğruluğunu doğrulamalıdır.

- Hedeflenen İşlemlerin Gerçekleşip Gerçekleşmediğinin Kontrol Edilmesi: Sistemde gerçekleştirilen işlemler, projenin hedeflerine ve gereksinimlerine uygun olmalıdır. Bilgi sistemleri denetçisi, bu işlemleri düzenli olarak gözden geçirerek gereksinimlerin karşılandığını doğrulamalıdır.

- Değişikliklerin Uyumunun Kontrol Edilmesi: Sistemde yapılan herhangi bir değişiklik, tasarım ve gereksinimlere uygun olmalıdır. Bilgi sistemleri denetçisi, değişikliklerin bu uyuma sahip olduğunu ve sistemin istenilen sonuçları verdiğini teyit etmelidir.

- Operatörlerin Hata Günlüklerinin İncelenmesi: Operatörlerin hata günlükleri, sistemdeki potansiyel sorunları ve hataları tespit etmek için önemlidir. Bilgi sistemleri denetçisi, bu günlükleri düzenli olarak inceleyerek sistemdeki kaynak veya işletme sorunlarını belirlemelidir.

Modern teknolojiler ve yöntemler, yazılım bakım süreçlerini daha verimli, hızlı ve güvenli hale getirmek için önemli bir rol oynamaktadır. Bu teknolojiler, geleneksel bakım yaklaşımlarını geliştirerek daha proaktif, otomatikleştirilmiş ve esnek çözümler sunar. Aşağıda, bu modern teknolojilerin ve yöntemlerin yazılım bakımına nasıl entegre edildiği ve bakım süreçlerini nasıl dönüştürdüğü açıklanmaktadır:

1. DevOps (Geliştirme ve Operasyonlar) Yaklaşımı ve Sürekli Entegrasyon (CI/CD) DevOps (Geliştirme ve Operasyonlar), yazılım geliştirme (Development) ve operasyonlar (Operations) arasındaki engelleri ortadan kaldıran bir yaklaşımdır. Bu, yazılım geliştirme sürecinin her aşamasında sıkı bir işbirliği ve otomasyon gerektirir. DevOps yaklaşımının temel amacı, yazılım geliştirme ve dağıtım süreçlerini hızlandırmak, kaliteyi artırmak ve hataları hızlı bir şekilde tespit etmektir.

Sürekli Entegrasyon (CI): Sürekli Entegrasyon, yazılım geliştirme sürecinde geliştiricilerin kodlarını sürekli olarak ana yazılım dalına entegre etmelerini sağlar. Bu, yazılımın her güncellemesinin, testlerden geçmeden ve kullanıcıya sunulmadan önce doğruluğunun kontrol edilmesini sağlar. CI araçları, yazılımın sürekli olarak test edilmesini ve hataların hızlıca tespit edilmesini sağlar. Bu yöntem, yazılımın her yeni sürümünde hataların daha erken tespit edilmesini sağlar ve yazılımın daha güvenilir hale gelmesine olanak tanır.

Sürekli Dağıtım (CD): Sürekli Dağıtım, yazılımın her başarılı testten sonra otomatik olarak üretim ortamına gönderilmesini sağlar. Bu otomasyon, yazılım güncellemelerinin daha hızlı ve sorunsuz bir şekilde yapılmasını mümkün kılar. Sürekli Dağıtım araçları, hata tespiti sürecini hızlandırır ve değişikliklerin anında kullanıcıya sunulmasını sağlar.

DevOps (Geliştirme ve Operasyonlar) ve CI/CD'nin sistem bakımına katkıları şu şekildedir:

Bakım Süreçlerinin Hızlandırılması: DevOps (Geliştirme ve Operasyonlar) uygulamaları, yazılım bakım süreçlerini hızlandırarak bakım işlemlerinin daha verimli hale gelmesini sağlar. Bu yaklaşım, sürekli test ve sürekli dağıtım sayesinde yazılım güncellemelerinin daha kısa sürede yapılmasına yardımcı olur.

Hızlı Hata Tespiti ve Çözüm: CI/CD araçları, yazılımın her aşamasında testlerin yapılmasını sağlar, böylece hatalar daha erken tespit edilir ve çözülür. Bu, bakım süreçlerinin etkinliğini artırır ve yazılımın güvenliğini sağlar.

2. Yapay Zeka (AI) ve Otomasyon: Yapay zeka (AI) tabanlı araçlar, sistem bakım süreçlerini daha proaktif hale getirir. AI, potansiyel sorunları tahmin ederek bakım ekibine erken uyarılar gönderir. Bu teknolojiler, bakım süreçlerinde zaman kaybını azaltarak daha hızlı çözüm üretir.

Proaktif Sorun Çözümü: Yapay zeka (AI) tabanlı araçlar, bakım sırasında oluşabilecek sorunları tahmin eder ve bunlar için otomatik çözüm önerileri sunar. Bu, bakım ekiplerinin sorunları daha erken fark etmelerini ve önlem almalarını sağlar.

Otomatik Hata Tespiti: Yapay zeka (AI), sistemdeki anormallikleri ve hataları tespit ederek bunları hızla tanımlar ve gerekli düzeltmeleri önerir. Bu süreç, manuel müdahaleyi azaltarak bakım sürecini daha verimli hale getirir.

3. Bulut Tabanlı Sistemler: Bulut bilişim, sistem bakımını merkezi bir platformda toplayarak daha hızlı ve verimli bir bakım süreci sağlar. Bulut tabanlı sistemlerde, veri yedekleme, güncelleme ve bakım işlemleri daha hızlı ve esnek bir şekilde yapılabilir.

Merkezi Bakım: Bulut sistemleri, bakım işlemlerini tek bir merkezden yönetmeye olanak tanır. Bu, sistemin her yerden izlenmesi ve bakımlarının yapılması için büyük bir kolaylık sağlar.

Veri Yedekleme ve Güncellemeler: Bulut tabanlı sistemlerde, verilerin düzenli olarak yedeklenmesi ve yazılım güncellemeleri merkezi olarak yapılabilir. Bu da, bakım sürecini daha hızlı ve verimli hale getirir.

4. Siber Güvenlik ve Sürekli İzleme: Siber güvenlik, sistem bakım süreçlerinin kritik bir parçasıdır. Sistemlerin güvenliğini sağlamak için sürekli izleme araçları kullanılır. Bu araçlar, potansiyel tehditleri tespit eder ve bu tehditlere hızlıca müdahale edilmesini sağlar.

Güvenlik İzleme: Sürekli güvenlik izleme araçları, sistemdeki potansiyel güvenlik tehditlerini izler ve bu tehditler hakkında bakım ekibini bilgilendirir. Bu, sistemin güvenliğini sağlamak ve hızlı müdahale için önemlidir.

Anormallik Tespiti: Güvenlik izleme araçları, sistemdeki anormal davranışları tespit eder ve hızlıca uyarılar gönderir. Bu da güvenlik açıklarını tespit etmeyi ve engellemeyi kolaylaştırır.

5. Proaktif Bakım ve İzleme Araçları: Proaktif bakım, sistemdeki potansiyel sorunları önceden tespit etmek ve çözüm önerileri sunmak amacıyla kullanılır. Sürekli izleme araçları, sistemdeki anormallikleri izler ve bakım ekiplerine hızlı uyarılar gönderir.

Gerçek Zamanlı İzleme: Proaktif bakım araçları, sistemin her anını izler ve anormal durumları anında tespit eder. Bu araçlar, bakım ekibine erken uyarılar göndererek sorunları çözmeden önce önlem almayı sağlar.

Hızlı Yanıt Süresi: Proaktif izleme araçları, hızlı yanıt süreleriyle bakım ekiplerinin sorunları hızla çözmesine olanak tanır. Bu, bakım süreçlerini hızlandırır ve sistemin istikrarlı çalışmasını sağlar.

Talep/Hata Yönetimi:

Gereksinimlerden elde edilen isterlere göre test senaryoları oluşturulur. Test senaryolarının sonuçları beklenen ile gerçekleşen arasında uyumsuzluk varsa hata olarak değerlendirilebilir. Karşılaşılan bu hatalar ile ilgili bir araç yardımıyla (Jira), sözlü ya da başka bir yöntem ile ilgililerine iletilmeli ve raporlanmalıdır. İletilen hatalar incelenir ve düzeltmeleri sağlandıktan sonra teste tabi tutulur. Hataya özel "*onay testi*" ve sistemde diğer alanları etkileyip etkilemediğini kontrol etmek için "*regresyon testi*" gerçekleştirilir. Hata raporları oluşturulurken, test araçları ya da dökümanlardan yararlanılabilir. Hataların tamamı giderilerek bitiş kriterlerine uygun hale gelmesi sonrasında test sonuç raporları hazırlanır.

Talep/Hata yönetimi süreci ile ilgili olarak aşağıdaki adımlar takip edilir:

- Test Senaryolarının Oluşturulması: Gereksinimlerden elde edilen isteklere dayalı olarak test senaryoları hazırlanır. Bu senaryolar, sistemin işlevselliğini, güvenliğini ve bütünlüğünü doğrulamak için kullanılır.

- Hataların Tanımlanması: Test senaryolarının sonuçları incelenir ve beklenen sonuç ile gerçekleşen sonuç arasındaki farklar hatalar olarak tanımlanır. Bu hatalar, bir araç (örneğin Jira) kullanılarak kayıt altına alınır.

- Hata Raporlarının İletilmesi: Tanımlanan hatalar, ilgili paydaşlara bildirilir. Bu bildirimler sözlü iletişim, yazılı iletişim veya diğer iletişim yöntemleriyle gerçekleştirilebilir.

- Hataların İncelenmesi ve Düzeltmelerin Sağlanması: İletilen hatalar, ilgili ekipler tarafından incelenir. Gerekli düzeltmeler yapılır ve hatalar giderilir.

- Test Sürecinin Tekrarlanması: Düzeltmeler yapıldıktan sonra, hataların tekrar test edilmesi gerekir. Bu, hataya özel "*onay testi*" ve sistemin diğer alanlarını etkileyip etkilemediğini kontrol etmek için "*regresyon testi*" içerebilir.

- Hata Raporlarının Dökümante Edilmesi: Hata raporları, test araçları veya dökümanlar kullanılarak kayıt altına alınır. Bu raporlar, hata tespiti ve düzeltme süreçlerini izlemek ve belgelendirmek için kullanılır.

- Hataların Giderilmesi ve Test Sonuç Raporlarının Hazırlanması: Tüm hatalar giderildikten sonra, test sonuç raporları hazırlanır. Bu raporlar, testlerin sonuçlarını, yapılan düzeltmeleri ve test kriterlerine uygunluğu özetler.

a. Hataların Önem Derecesi (Severity)

Testlerde esnasında tespit edilen yazılım hatasının önem derecesi, sistemin çalışmasındaki hata etkisinin derecesini belirler. Tespit edilen hatalar *önem derecesine* göre ayrılır.

Düşük (Low) –> Testlerde ilerlemeye etkisi olmayan minör olarak değerlendirilen hatalardır. Test edilen öğe bu hatalar ile teslim edilebilir.

Orta (Medium) --> Testler devam eder. Bu hatalar ile teslim edilen ürün, daha sonra düzeltilebilecek hatalar içerebilir. Sistemin çalışmasını bozmayan hatalardır.

Yüksek (High) –> Sistemin çalışmasına etki eden fakat testlere devam etmeye engel olmayan hatalardır. Düzeltilmesi öncelikli olarak değerlendirilecek hatalardır.

Kritik (Critical) –> Testleri engelleyen ve çözülmesi mutlaka gerekli olan hatalardır.

b. Hataların Öncelik Seviyesi (Priority)

Testlerde tespit edilen hatalar analiz edilerek bir önceliklendirme yapılır ve karşılaşılan hatalar için öncelik seviyeleri belirlenir. Genel olarak testlerin devam etmesini engelleyen kritik seviyedeki hatalar için öncelik verilir. Hatalar için öncelik seviyesi aşağıdaki gibi olabilir.

Düşük (Low) –> Sistemin çalışmasına etki yaratmayan hatalardır.

Orta (Medium) –> Sistemin çalışmasını engellemeyen ancak düzeltilmesi gerekli ve teslimden sonra düzeltilebilecek hatalardır.

Yüksek (High) –> Sistemin işleyişini durdurmayan ama çalışmasını etkileyen hatalardır.

Kritik (Critical) –> Sistemin işleyişini ve çalışmasını engelleyen hatalardır. Düzeltilmesi acil ve önemli hatalardır.

Gereksinimlere göre hazırlanan test senaryolarında beklenen ile gerçekleşen durumların uyumsuz olduğunda oluşan hataları yukarıda Önem ve öncelik seviyelerine göre hata detayları oluşturulabilir. Proje ya da talep kapsamında hata açılacak ise belirtilmelidir. Hata özeti açık ve anlaşılır olacak şekilde eklenir.

Yazılım geliştirme süreçlerinde, hata yönetimi ve test senaryolarının etkin bir şekilde yönetilmesi, yazılımın kalitesini ve güvenilirliğini doğrudan etkileyen faktörlerdir. Bu bağlamda, yeni teknolojilerin entegrasyonu, hata tespitini hızlandırır, raporlama süreçlerini optimize eder ve yazılımın kalitesini artırır. Aşağıda, hata yönetimi sürecine yönelik modern yaklaşımlar ve araçlar detaylı bir şekilde ele alınmıştır. Bu araçlar, yazılım geliştirme süreçlerinde daha hızlı ve daha verimli test yapmayı mümkün kılmakta, yazılımın sürekli olarak güncel ve hatasız kalmasını sağlamaktadır. Aşağıda hata yönetimi sürecine yönelik bazı modern yaklaşımlar ve araçlar açıklanmaktadır:

Test Otomasyonu ve Hata Yönetimi İlişkisi: Test otomasyonu, hata yönetim süreçlerinin hızlandırılmasında önemli bir rol oynamaktadır. Test senaryoları oluşturulurken, bu senaryoların

otomatikleştirilmesi, testlerin tekrarlanabilirliğini ve doğruluğunu artırır. Otomatikleştirilmiş testler, manuel testlerin aksine, hataların daha hızlı ve tutarlı bir şekilde raporlanmasını sağlar. Özellikle büyük yazılım projelerinde, otomatik test araçları (Jenkins, Selenium, vs.) kullanılarak test edilen yazılımda hataların hızlıca belirlenmesi sağlanır. Ayrıca, test senaryoları her değişiklikten sonra otomatik olarak çalıştırılır ve herhangi bir uyumsuzluk veya hata durumunda hızla hata raporu oluşturulabilir.

Yapay Zeka ve Makine Öğrenmesi ile Hata Tahmini: Gelişen yapay zeka ve makine öğrenmesi (ML) teknolojileri, yazılım geliştirme ve test süreçlerine yeni bir boyut kazandırmaktadır. Yapay zeka, geçmiş test sonuçlarından ve hatalardan öğrenerek, gelecekte oluşabilecek potansiyel hataları tahmin etme yeteneğine sahiptir. Bu tür AI tabanlı hata tahmin araçları, test senaryolarını ve hata yönetim süreçlerini proaktif hale getirir. Makine öğrenmesi algoritmaları, yazılımın önceki sürümlerinde hangi hataların daha sık tekrarlandığını analiz edebilir ve bu hataların gelecekteki sürümlerde tekrar etmesini önlemek için gerekli tedbirleri önerebilir.

Hata Yönetimi için Analitik Araçlar: Hata yönetim süreçlerinde analitik araçlar kullanımı, yazılım geliştirme süreçlerinin izlenmesi ve yönetilmesi açısından önemlidir. Analitik araçlar, hataların türleri, tekrarlama oranları ve hata tespit süresi gibi metrikleri toplar ve raporlar. Bu araçlar, hataların ne kadar süreyle çözülmesi gerektiğini, hangi hataların daha öncelikli olduğunu ve yazılımın hangi bölümlerinin en fazla hata ürettiğini gösterir. Bu bilgiler, yazılım geliştirme ekibinin kaynaklarını daha verimli kullanmasını ve hataların daha hızlı çözülmesini sağlar.

Sürekli Entegrasyon (CI) ve Hata Yönetimi Süreci: Sürekli entegrasyon (CI) süreçlerinde, her yazılım değişikliği anında test edilir ve hatalar hızla belirlenir. CI/CD (Sürekli Entegrasyon ve Sürekli Dağıtım) araçları, her kod güncellemesinden sonra otomatik olarak testler çalıştırarak, yazılımın tüm bileşenlerinin uyumlu çalıştığından emin olur. Bu süreç, hataların geliştirme aşamasında tespit edilmesini sağlar ve yazılımın her an güncel ve hatasız olmasını garanti eder. Ayrıca, CI süreçlerinde yazılımın tüm hataları, testler sırasında otomatik olarak raporlanır ve geliştirme ekibine iletilir.

c. Hata Kaydı Açılması

Hata kaydı açılması aşamasında, hatalar için öncelik seviyesi belirlenir ve hata açıklaması detaylı ve açık bir şekilde girilir. Hata açıklamasında test senaryosunda kullanılan data, ortam bilgileri de girilmelidir. Test senaryosu girilir. Beklenen ve gerçekleşen durum belirtilir. Ekran görüntüsü ve alınabilirse video ile desteklenir. Hata oluşturulur ve düzeltecek kişi ataması yapılır. Hatalar takip edileceği, JIRA (Atlassian tarafından geliştirilen ve hata takibi ve çevik proje yönetimi sağlayan hata izleme aracı), TestRail, HP ALM gibi araçlardan faydalanılabilir ya da excelde tablo oluşturularak mail yoluyla geliştiricilere bildirilir.

Testler sırasında karşılaşılan hataların açılması ile düzeltilmesine kadar süreç aşağıdaki gibi izlenebilir. Testleri yapan çalışanlar tarafından tespit edilen hataların öncelikle önem derecesi belirlenir. Kullanılıyorsa bir araç yardımıyla Hata kaydı oluşturulur. Hatayı çözecek kişiye atanır. Testleri engelleyen durum olması halinde çözüm beklenir. Engelleyen bir durum yoksa diğer test senaryolarına devam edilir. Hatanın atandığı kişi çözüm için çalışma başlatır. Analiz yapılır. Hata olduğu kabul edilmeyen durumlarda yazılımcı tarafından hata reddedilir. Hatanın kabul edilerek düzeltildiği durumda tekrar test yapılması için testi yapan kişiye atanır. Hata düzeltilmiş ve testi başarılı olan hata kaydı kapatılır. Test başarısız sonuçlanırsa hata kaydı yeniden açılır ve düzeltilmesi için süreç yeniden yönetilir.

Hata kaydı izleme ve takip süreci, projenin hata yönetimi ve düzeltme sürecinin etkin bir şekilde yönlendirilmesini sağlar. Bu süreç içerisinde aşağıdaki adımların uygulanması faydalı olacaktır:

- Hata Kaydı Takibi: Hata kaydı, projenin hata izleme ve yönetim araçları (örneğin JIRA, TestRail, HP ALM) üzerinden takip edilir. Bu sayede hata kaydının ne zaman açıldığı, kim tarafından atandığı, düzeltilme süreci, atama değişiklikleri ve kapanma durumu gibi bilgilere erişilebilir.

- Hata Çözüm Süreci: Hata, atanmış olan kişi veya ekibi tarafından çözülür. Çözüm sürecinde analiz yapılır, düzeltme gerçekleştirilir ve gerekirse revizyonlar yapılır. Hata çözümü tamamlandığında, bu bilgi hata kaydı üzerinde güncellenir.

- Tekrar Testi ve Onay: Hata çözüldükten sonra, test ekibi tarafından tekrar test edilir. Eğer hata düzeltilmişse ve testler başarılı sonuçlanırsa, hata kaydı kapatılır ve hata artık düzeltilmiş olarak kabul edilir.

Program Değişikliklerinin Denetimi ve Acil Değişiklikler:

Bilgi sistemleri denetçisi tarafından, uygulamanın tamamlanması sonrasında aşağıdaki hususları dikkate almalıdır:

- Acil durum değişiklikleri ile ilgili prosedürlerin yeterliliği ve uygulama yönetimi,

- Kullanıcı taleplerine yönelik gerçekleştirilen geri dönüşler (zaman ve maliyet) için memnuniyet ölçüm kayıtları,

- Güvenlik erişim kısıtlamalarının, üretim kaynak ve çalışabilirlik modülleri üzerindeki etkinliği ve yeterliliği,

- Acil durum değişiklik yönetimine ait prosedürleri,

- Acil durum oturum açma kimlikleri ile ilgili güvenlik erişim kısıtlamalarının yeterliliği,

- Değişiklik kontrollerinin, kullanıcı ve geliştirici grupları için yeterli bir prosedürün olup olmadığının kontrol edilmesi,

- Kullanıcı değişiklik taleplerinde yetkilendirme, önceliklendirme ve izlemek için önceden belirlenmiş metodolojilerin kullanımı ve yeterliliğinin değerlendirilmesi,

- Değişiklik kontrol günlüğünde gösterilen tüm değişikliklerin çözülüp çözülmediğinin kontrol edilmesi.

Bilgi sistemleri denetçisi, onaylama, müdahale süresi, yanıt etkinliği ve süreçten kullanıcının memnuniyeti konusundaki olası iyileştirmeler için tüm değişim yönetimi surecini gözden geçirmelidir.

Acil değişiklik yönetimi ve süreç iyileştirmeleri, aşağıdaki şekilde ele alınmalı ve etkin bir şekilde yönetilmelidir:

Acil değişiklikler, genellikle aniden ortaya çıkan sorunlara hızlıca çözüm sağlamak amacıyla yapılır. Bu tür değişiklikler, özellikle kritik sistem hatalarına karşı acil müdahale gerektiren durumlarda ortaya çıkar. Ancak, acil değişikliklerin yönetimi, dikkatli planlama ve izleme gerektiren bir süreçtir. Acil değişikliklerin etkili bir şekilde yönetilmesi, yalnızca sistemin güvenliğini sağlamakla kalmaz, aynı zamanda organizasyonun genel iş süreçlerine olan etkisini de minimize eder.

Acil değişiklikler, genellikle sistemdeki yüksek riskleri hedef alırken, hızlı çözüm üretmeyi amaçlar. Bu, değişikliklerin nasıl denetleneceği ve hangi prosedürlerin uygulanacağı konusunda açık bir çerçeve gerektirir. Acil değişiklik prosedürleri, genellikle şu unsurları içerir:

- Acil Durum Değişikliklerinin Kapsamı: Değişikliğin gerekli olduğu durumlar ve koşullar, önceden belirlenmiş olmalıdır. Değişiklikler yalnızca kritik hatalar veya güvenlik açıkları gibi durumlar için yapılmalıdır.

- Acil Müdahale Süreçlerinin Belirlenmesi: Acil değişikliklere müdahale için özel süreçler belirlenmeli, süreçlerin hızla uygulanabilmesi için gereksinimler netleştirilmelidir. Bu süreçler, sistemdeki kesintilerin minimize edilmesini sağlayacak adımları içerir.

- Zaman Kısıtlamaları ve Yöneticilerle Koordinasyon: Acil değişiklikler yapılırken zaman kısıtlamaları önemli rol oynar. Bu değişikliklerin hızlıca yapılması için uygun kaynakların ayrılması ve yöneticilerle koordinasyon sağlanmalıdır.

- Geriye Dönüş Senaryoları ve Yedekleme: Acil bir değişiklik uygulandığında, olası hataların telafi edilebilmesi için geri dönüş senaryolarının ve yedekleme planlarının uygulanması kritik öneme sahiptir.

Modern değişiklik yöntemleri ve araçları, aşağıda açıklanan şekilde etkili bir şekilde uygulanmalı ve entegrasyon sağlanmalıdır:

Günümüz yazılım geliştirme dünyasında, değişiklik yönetimi süreçlerinin etkinliğini artırmak için birçok yeni teknoloji ve araç kullanılmaktadır. Bu araçlar, değişikliklerin kaydedilmesi, izlenmesi ve geri bildirimlerin alınması süreçlerini hızlandırır ve şeffaf hale getirir.

Sürekli Entegrasyon (CI) ve Sürekli Dağıtım (CD), değişikliklerin hızlı bir şekilde test edilmesini ve dağıtılmasını sağlayan modern yazılım geliştirme yöntemlerindendir. Bu yöntem, yazılım değişikliklerinin üretim ortamına hızlı bir şekilde entegre edilmesini sağlar. CI/CD araçları, yazılım geliştirme sürecindeki değişikliklerin her aşamada izlenmesine olanak tanır ve hataların hızlıca tespit edilmesini sağlar. Bu süreç, yazılım bakımını hızlandırarak, bakım sürecine daha verimli bir yaklaşım getirir.

Değişiklik İzleme Araçları: Jira, TestRail ve HP ALM gibi araçlar, değişikliklerin izlenmesi ve kaydedilmesi süreçlerinde büyük kolaylık sağlar. Bu araçlar, hem geliştirme sürecindeki hataların hem de bakım aşamasındaki değişikliklerin düzgün bir şekilde yönetilmesini sağlar. Değişiklik izleme araçları, özellikle büyük projelerde, hata raporlarının ve düzeltmelerin organizasyonel düzeyde takip edilmesini sağlar. Bu araçlarla, her değişiklik kaydedilebilir ve bir izleme süreci başlatılabilir, böylece yazılım sürecindeki her değişiklik anında izlenebilir.

Proaktif Bakım Yöntemleri: Yazılımda yapılan her değişiklik, mevcut sistemle entegrasyonunun etkisini test etmek için düzenli olarak izlenmeli ve analiz edilmelidir. Proaktif bakım, yazılımın olası sorunlarını erkenden tespit ederek, bakım süreçlerini minimize etmeyi hedefler. Bu, sistem performansını artırırken, bakım maliyetlerini de azaltır.

Değişiklik yönetiminin sonuçlarının izlenmesi, aşağıda belirtilen yöntemlerle dikkatlice takip edilmeli ve değerlendirilmelidir:

Değişiklik yönetimi, yalnızca değişikliklerin yapılmasını değil, aynı zamanda bu değişikliklerin etkilerinin izlenmesini de içerir. Bilgi sistemleri denetçisi, yapılan değişikliklerin hedeflere ve gereksinimlere uygun olup olmadığını düzenli olarak izlemelidir. Değişiklik sonrası raporlama ve izleme, hem teknik ekiplere hem de yöneticilere bilgi sağlar, böylece sistemdeki potansiyel zayıflıklar hızla tespit edilebilir.

Özellikle, değişiklik sonrası testlerin ve geri dönüş senaryolarının etkinliği, değişikliklerin başarılı bir şekilde uygulanıp uygulanmadığını belirler. Bu nedenle, değişiklik yönetimi süreci, sadece yapılacak işlemleri değil, aynı zamanda bu işlemlerin izlenmesini ve raporlanmasını da kapsamlı bir şekilde ele almalıdır. Bu sayede, hem kullanıcıların memnuniyeti sağlanabilir hem de sistemin sürdürülebilirliği garanti altına alınır.

Sonuç olarak, program değişikliklerinin yönetimi, doğru araçlar, prosedürler ve yöntemlerle etkin bir şekilde yapılmalıdır. Hızlı ve doğru değişiklikler, bir sistemin güvenliğini, bütünlüğünü ve verimliliğini artırır. Bu süreçlerin etkin bir şekilde izlenmesi ve yönetilmesi, yazılımın sağlıklı bir şekilde işletilmesini sağlar ve kurumların hedeflerine ulaşmalarını kolaylaştırır.

Geri Dönüş Prosedürleri:

Başarılı tamamlanan testler sonrasında değişim kontrol prosedürlerine göre sistemin kuruluşu gerçekleştirilmelidir. Sistem kuruluşu yapılmadan önce kuruluş sürecinin nasıl yapılacağına ilişkin bir gerçekleştirme planı hazırlanmalıdır. Kuruluş süreç adımları tek tek belirlenmeli ve sorumluları eklenmelidir. Adımların nasıl doğrulanacağı ve geri dönüş planlarına ilişkin prosedürler oluşturulmalıdır. Burada geri dönüş senaryoların gerçekleştirilmesi çok önemlidir.

Uygulanan değişikliğin başarısız olması durumunda, geri döneceği sürümler (bilinen sonuçlar) referans alınmalıdır. Geri dönüş prosedürlerinin başarılı olması için, bu sürecin başarılı işletilebilmesi gerekir.

Bilgi sistemleri denetçisi tarafından kullanıcının değişiklik taleplerinin geri dönüşü (zaman ve maliyet) için memnuniyeti değerlendirilmesi gerekmektedir. Bununla birlikte, kullanıcının değişiklik taleplerinin geri dönüşü sadece zaman ve maliyet açısından değil, aynı zamanda kullanıcının taleplerine verilen yanıtın kalitesi ve uygunluğu açısından da değerlendirilmelidir. Kullanıcının memnuniyeti, sistemin kullanıcı odaklılığını ve iş ihtiyaçlarını ne kadar etkili bir şekilde karşıladığını gösteren önemli

bir göstergedir. Bu nedenle, değişiklik taleplerinin sonuçları kullanıcının beklentilerini karşılayıp karşılamadığını ve iş süreçlerine nasıl entegre olduğunu dikkate alarak değerlendirilmelidir.

Geri dönüş prosedürlerinin etkinliğini artırmak ve bu süreçleri daha verimli hale getirebilmek için modern yazılım geliştirme yaklaşımları ve araçları kullanılabilir. Bu araçlar, yazılımın her aşamasında geri dönüşlerin hızlı ve güvenli bir şekilde yapılmasını sağlamakla kalmaz, aynı zamanda sürecin izlenebilirliğini artırır. Bu bağlamda, yazılımın daha esnek ve hızlı geri dönüş senaryolarını uygulayabilir olması önemlidir.

Özellikle CI/CD (Sürekli Entegrasyon ve Sürekli Dağıtım) süreçleri, değişikliklerin hızlı bir şekilde üretim ortamına alınmasını sağlar. Bu süreç, herhangi bir olumsuz durumda hızlıca geri dönüş yapılabilmesi için gerekli altyapıyı sunar. Herhangi bir değişiklik, önce test ortamında izlenmeli ve sorun tespit edildiği takdirde anında geri alınabilmelidir. Geri dönüşler için yazılımın yedeklemeleri ve sürüm yönetimi de kritik bir öneme sahiptir. Sürüm kontrol sistemleri (örneğin Git, Bitbucket), her değişikliğin izlenebilir ve geri döndürülebilir olmasını sağlar, bu da sorunlar ortaya çıktığında sistemin önceden çalışan haline geri dönmeyi mümkün kılar.

Ayrıca, geri dönüş prosedürlerinin başarısı için doğru test stratejilerinin belirlenmesi gerekmektedir. Regresyon testi ve onay testi, yapılacak değişikliklerin sistemin diğer bölümlerine zarar vermediğini doğrulamak için kritik öneme sahiptir. Regresyon testi, özellikle yazılımın eski sürümünde düzgün çalışan fonksiyonların yeni sürümde de doğru çalışıp çalışmadığını kontrol eder. Bu testler, sistemdeki tüm işlevselliğin beklenildiği gibi devam ettiğinden emin olmak için gereklidir.

Proaktif bakım ve izleme de geri dönüş prosedürlerini destekler. Sistem üzerinde yapılan her değişiklik, yazılımın genel performansını ve kullanımını etkileyebilir. Bu nedenle, değişiklik sonrası yazılımın performansı izlenmeli ve olası sorunlar erkenden tespit edilmelidir. Proaktif izleme araçları, herhangi bir anormalliği erken dönemde tespit ederek, yazılımın orijinal duruma dönmesini sağlayacak adımların atılmasına olanak tanır.

Geri dönüş senaryolarının oluşturulması, sistemin kesintisiz çalışmasını sağlayan kritik bir unsurdur. Bu senaryolar, her tür değişikliğin olası etkilerini değerlendirerek, sistemdeki her modülün uygun şekilde geri alınabilmesini sağlar. Bu nedenle, geri dönüş prosedürlerinin ayrıntılı ve dikkatli bir şekilde planlanması, herhangi bir sistemsel hata durumunda kesintilerin minimuma indirilmesini sağlar.

Geri dönüş testlerinin yanı sıra performans izleme ve güvenlik değerlendirmeleri de önemlidir. Yapılan değişikliklerin sadece işlevsellik üzerindeki etkisini değil, aynı zamanda sistemin güvenliğini ve performansını nasıl etkilediğini de göz önünde bulundurmak gerekir. Bu sayede, yalnızca yazılımın düzgün çalışıp çalışmadığı değil, aynı zamanda güvenlik açıklarının oluşup oluşmadığı da tespit edilebilir.

Son olarak, kullanıcı geribildirimi ve memnuniyet analizi, değişikliklerin başarısını değerlendirmek için kritik veriler sağlar. Geri dönüş prosedürlerinin etkinliğini anlamak için kullanıcıların değişikliklere tepkileri incelenmeli ve bu tepkiler ışığında süreçte iyileştirmeler yapılmalıdır. Kullanıcı memnuniyeti, hem işlevsel hem de operasyonel açıdan sistemin verimliliğini ölçmek için en önemli kriterlerden biridir.

Uygulama Sonrası Gözden Geçirme ve Öğrenilmiş Dersler:

Proje işletme yararlarını ve maliyetlerini gerçekleştirmek, projenin genel başarısını ve iş birimleri üzerindeki etkisi görüldükten sonra gözden geçirme yapılabilir. Bu aşamada ölçütler aşağıdakileri içerir:

- Toplam sahip olma maliyeti (TCO): TCO, bir projenin yaşam döngüsü boyunca sahip olma ve işletme maliyetlerini hesaplamak için kullanılır. Bu maliyetler donanım, yazılım, bakım, personel maliyetleri ve diğer ilgili giderleri içerebilir. Projenin toplam sahip olma maliyeti, projenin uzun vadeli mali etkisini değerlendirmeye yardımcı olur.

Toplam Maliyet (TCO), bir projenin, bir ürünün veya bir hizmetin yaşam döngüsü boyunca toplam maliyetini hesaplamak için kullanılan bir kavramdır. Genel TCO formülü aşağıdaki gibidir:

TCO = Maliyetler + İşletme ve Bakım Maliyetleri + Sonlandırma Maliyetleri

Maliyetler: İlk yatırım maliyetleri, satın alma maliyetleri, lisans maliyetleri gibi başlangıç maliyetlerini ifade eder. Bunlar projenin veya ürünün kurulum aşamasında ortaya çıkan maliyetlerdir.

İşletme ve Bakım Maliyetleri: Ürün veya projenin işletilmesi ve bakımı için gereken maliyetleri içerir. Bu, enerji tüketimi, personel maaşları, bakım ve onarım maliyetleri gibi sürekli işletme maliyetlerini içerir.

Sonlandırma Maliyetleri: Ürün veya proje sona erdiğinde veya yeni bir sürümü devreye alındığında, mevcut sistemi sonlandırma veya geçişi yönetme maliyetlerini içerir.

TCO hesaplamaları, bir projenin veya yatırımın gerçek maliyetini daha iyi anlamak ve uzun vadeli maliyetleri tahmin etmek için kullanılır. Bu hesaplamalar, karar vericilere bütçeleme, kaynak tahsisi ve proje değerlendirmesi konularında yardımcı olabilir.

- Yatırım getirisi (ROI): ROI, bir projenin maliyetine karşı elde edilen faydaları değerlendirmek için kullanılır. Genellikle finansal bir oran olarak ifade edilir ve projenin yatırımını ne kadar sürede geri kazanabileceğinizi gösterir. Yüksek bir ROI, projenin maliyetine göre daha fazla fayda sağladığını gösterir.

ROI'nin temel formülü aşağıdaki gibidir:

ROI= (Net Kazanç/Toplam Yatırım Maliyeti)*100

Bu formüldeki bileşenler aşağıdaki gibi açıklanabilir:

Net Kazanç: Yatırımın getirdiği gelirlerin, yatırımın toplam maliyetlerinden çıkarılması ile elde edilen karı temsil eder.

Toplam Yatırım Maliyeti: Yatırımın başlangıçta yapılan tüm maliyetlerini içerir. Bu maliyetler, satın alma maliyetleri, kurulum maliyetleri, işletme ve bakım maliyetleri, sonlandırma maliyetleri gibi çeşitli unsurları içerebilir.

ROI, yatırımın ne kadar karlı olduğunu değerlendirmek için kullanılır. Bir yatırımın ROI değeri pozitifse, bu yatırımın kar getirdiği anlamına gelir. Negatif bir ROI ise yatırımın maliyetlerini karşılayamadığını veya kayıplara neden olduğunu gösterir.

ROI hesaplaması, finansal kararlar alırken yatırımcılara ve işletmelere yardımcı olabilir. Yatırım fırsatlarını değerlendirmek, projelerin karlılığını izlemek ve kaynakları etkili bir şekilde yönlendirmek için yaygın bir metriktir.

Aşağıda toplam sahip olma maliyeti (TCO) ve yatırım getirisi (ROI) ile ilgili örnek hesaplamalara yer verilmiştir:

Örnek 1: Yazılım Yükseltmesi

Bir şirket, mevcut yazılımını güncellemeyi planlıyor. Güncelleme, yazılımın lisans maliyeti ve personel eğitimi gerektiriyor.

Maliyetler:

Yazılım lisansları: 20.000 TL

Personel eğitimi: 5.000 TL

Toplam Maliyet = 20.000 TL + 5.000 TL = 25.000 TL

Fayda:

Güncel yazılımın getireceği iş verimliliği artışı: 40.000 TL

Toplam Fayda = 40.000 TL

TCO = 25.000 TL

ROI = ((40.000 TL - 25.000 TL) / 25.000 TL) * 100 = 60%

Bu durumda, yazılım güncellemesi TCO açısından 25.000 TL'ye mal olurken, ROI açısından %60'lık bir getiri sağlar.

Örnek 2: Ekipman Yatırımı

Bir restoran, mutfak ekipmanlarını yenilemeyi düşünüyor. Yeni ekipmanlar, mevcut ekipmanların yerine geçecek.

Maliyetler:

Yeni ekipmanlar: 50.000 TL

Eski ekipmanların satışı: 10.000 TL

Toplam Maliyet = 50.000 TL - 10.000 TL = 40.000 TL

Fayda:

Yeni ekipmanların verimliliği artışı: 30.000 TL

Toplam Fayda = 30.000 TL

TCO = 40.000 TL

ROI = ((30.000 TL - 40.000 TL) / 40.000 TL) * 100 = -25%

Bu durumda, ekipman yatırımı TCO açısından 40.000 TL'ye mal olurken, ROI açısından %25'lik bir kayba neden olur.

Örnek 3: Eğitim Yatırımı

Bir birey, kariyerinde ilerlemek için bir eğitim programına katılmayı düşünüyor. Eğitim programının maliyeti 7.500 TL ve programı tamamladıktan sonra daha yüksek bir maaş ve yeni yetenekler kazanmayı hedefliyor.

Maliyet:

Eğitim programı: 7.500 TL

Toplam Maliyet = 7.500 TL

Fayda:

Maaş artışı ve yeni yeteneklerin getireceği kariyer fırsatları: 20.000 TL

Toplam Fayda = 20.000 TL

TCO = 7.500 TL

ROI = ((20.000 TL - 7.500 TL) / 7.500 TL) * 100 = 166.67%

Bu durumda, eğitim yatırımı TCO açısından 7.500 TL'ye mal olurken, ROI açısından %166,67'lik bir getiri sağlar.

Örnek 4: Fabrika Otomasyonu

Maliyetler:

Yeni otomasyon ekipmanları: 150.000 TL

Personel eğitimi: 10.000 TL

Toplam Maliyet = 150.000 TL + 10.000 TL = 160.000 TL

Fayda:

Üretim verimliliğindeki artış ve hata oranının azalması: 75.000 TL

Toplam Fayda = 75.000 TL

TCO = 160.000 TL

ROI = ((75.000 TL - 160.000 TL) / 160.000 TL) * 100 = -53.13%

Bu durumda, fabrika otomasyonu TCO açısından 160.000 TL'ye mal olurken, ROI açısından %53,13'lük bir kayba neden olur.

Örnek 5: Yazılım Geliştirme Projesi

Maliyetler:

Yazılım geliştirme ekipmanları ve lisansları: 80.000 TL

Proje yönetimi ve test maliyetleri: 15.000 TL

Toplam Maliyet = 80.000 TL + 15.000 TL = 95.000 TL

Fayda:

Yeni yazılımın sağlayacağı iş süreçlerinde hızlanma ve hata azalması: 120.000 TL

Toplam Fayda = 120.000 TL

TCO = 95.000 TL

ROI = ((120.000 TL - 95.000 TL) / 95.000 TL) * 100 = 26.32%

Bu durumda, yazılım geliştirme projesi TCO açısından 95.000 TL'ye mal olurken, ROI açısından %26,32'lik bir getiri sağlar.

Bilgi sistemleri denetçisi tarafından uygulama sonrası gözden geçirme sırasında aşağıdaki hususları dikkate almalıdır:

- Kontroller gözden geçirilmeli ve tasarıma göre çalıştığından emin olunmalı,

- Yönetime doğru raporların ulaşıp ulaşmadığını teyit etmek için, maliyet faydalarının analizlerinin kontrol edilmeli,

- Operatörlerin oluşturduğu hata günlükleri incelenerek, kaynak veya işletme sorunu yaşanıp yaşanmadığı tespit edilmeli,

- Sistemin hedeflerine ve gereksinimlerine ulaşılıp ulaşılmadığı kontrol edilmeli,

- Giriş ve çıkış kontrol raporlarını gözden geçirerek, sistemin doğru çalıştığına emin olunmalı,

- Gerçekleştirilen değişikliklerin gözden geçirilerek, gereksinim duyulan değişiklikler ile uyumunun teyit edilmeli.

2.3. Uygulama Kontrolleri

Bilgi sistemleri süreç ve ortamlarında kullanılan kontroller genel kontroller ve uygulama kontrolleri olarak ikiye ayrılır. Uygulama kontrolleri ilgili iş süreçlerinin bilgi sistemlerine, girişinden çıkışına kadar, organizasyonel hedeflere katkı sağlayarak tüm süreç boyunca tam ve doğru bir şekilde işlevselliği yerine getirmesi amacıyla tasarlanmış kontrollerdir.

Verilerin sistemlere girilmesiyle başlayan uygulama kontrolleri; yetkilendirmelerin, gizlilik, bütünlük ve tutarlılıklarının kontrol edilmesi, oluşan hataların tespit edilmesi ve raporlanması, veri işleme, hesaplama, karşılıklı anlaşmaların sağlanması ve oluşan çıktıların kontrol edilmesi gibi kritik kontroller sağlar. Uygulama kontrolleri ile aynı zamanda hatasız ve planlanan sonuca ulaşmak sağlanır. İstisnai durumlar ve gerçekleşen hataların kök nedenlerinin araştırılarak otomatik kontroller ile prosedürlerin birleştirilmesi ile aksiyonlar alınabilir.

Uygulama kontrolleriyle;

- Bilgisayar sistemlerine sadece doğru, geçerli ve tam olacak şekilde verilerin girilmesi,

- Belirlenmiş olan doğru görevleri yerine getirmesi,

- Kayıt altına alınan verilerin korunması,

- Gerçekleşen sonuçların beklentileri karşılaması,

sağlanır.

Uygulama kontrolleri 3 gruba ayrılır. Bunlar:

- Girdi kontrolleri,

- İşlem kontrolleri,

- Çıktı kontrolleri.

Uygulama kontrolleri kapsamında bilgi sistemleri denetçisi aşağıdaki hususlara dikkat etmelidir:

- Verimlilik ve etkinliğin sağlanabilmesi için uygulamalardaki operasyonel taraflara dikkat etmek,

- Kritik uygulama bileşenlerini ve iş adımlarını belirlemek,

- Test stratejisi belirlemek,

- Uygulama kontrol etkinliğini ve oluşan kontrol zayıflıklarının etkisini incelemek,

- Etkinliğin sağlanabilmesi için kontrolleri test etmek,

- Planlanan kontrol hedeflerine ulaşılma başarısını tespit edebilmek için test sonuçlarını ve diğer denetim kanıtlarını analiz etmek.

Uygulama kontrolleri, bilgi sistemlerinin doğruluğunu, güvenliğini ve etkinliğini sağlamak için kritik bir rol oynar. Bu kontrollerin temel amacı, girişten çıktıya kadar tüm süreçlerin doğru, geçerli ve güvenli bir şekilde işlemesini temin etmektir. Uygulama kontrollerinin başarılı bir şekilde uygulanması, bilgi sistemlerinin güvenliği ve verimliliği açısından önemlidir.

Uygulama kontrolleri, sadece veri girişlerinin doğruluğunu sağlamakla kalmaz, aynı zamanda iş süreçlerinin her aşamasındaki işlevlerin doğru ve güvenli bir şekilde çalışmasını kontrol eder. Bu bağlamda, girdi kontrolleri, işlem kontrolleri ve çıktı kontrolleri olmak üzere üç ana kategoride uygulanır. Her bir kontrol türü, farklı aşamalarda sistemin doğru çalışmasını ve işlevselliğini sağlamak için kritik öneme sahiptir.

Girdi Kontrolleri: Bu kontroller, sisteme veri girişinin doğruluğunu ve güvenliğini sağlamak amacıyla yapılır. Veri girişlerinin doğru, geçerli ve eksiksiz olmasını temin etmek için çeşitli prosedürler kullanılır. Örneğin, kullanıcıların yalnızca yetkilendirilmiş verileri girmeleri için sistemde erişim kısıtlamaları ve doğrulama mekanizmaları yerleştirilebilir. Girdi kontrolleri, hatalı verilerin sisteme girmesini engelleyerek sistemin başlangıcındaki potansiyel hataları önler.

İşlem Kontrolleri: Veri girişinin ardından, sistemin her adımda doğru işlediğinden emin olmak için işlem kontrolleri yapılır. Bu kontroller, verilerin işlenmesinin doğru bir şekilde gerçekleştirildiğini ve her adımda istenen işlemin yapıldığını doğrular. Özellikle, sistemdeki işlemler karmaşıklaştıkça, her işlem adımının doğru şekilde kontrol edilmesi gerekir. Bu tür kontroller, hesaplamaların doğruluğunu, veri işleme süreçlerinin etkinliğini ve iş akışlarının doğru bir şekilde takip edilmesini sağlar.

Çıktı Kontrolleri: Sistemin çıktılarının doğruluğunu ve geçerliliğini denetleyen kontrol türüdür. Çıktı kontrolleri, sistemin nihai ürünlerinin veya verilerinin hedeflere uygunluğunu değerlendirir. Bu çıktılar, organizasyonel hedeflerle uyumlu olmalı ve belirli kalite standartlarına uygun şekilde düzenlenmelidir. Çıktı kontrolleri, sistemdeki hataların son kullanıcıya yansımadan önce tespit edilmesini sağlar.

Uygulama kontrolleri sadece yazılımın güvenliğini sağlamaz, aynı zamanda organizasyonun iş süreçlerini destekler. Prosedürel kontroller ve otomatik kontroller arasındaki entegrasyon da bu noktada önem kazanır. Otomatik sistemler, hataların hızlı bir şekilde tespit edilmesini ve gerektiğinde müdahale edilmesini sağlar. Ayrıca, istisnai durumların yönetimi ve hataların kök nedenlerinin belirlenmesi de bu tür kontroller aracılığıyla yapılabilir. Prosedürel kontroller, organizasyonel gereksinimleri karşılayacak şekilde daha manuel bir kontrol süreci sunar.

Test Stratejisi ve Denetim Süreci: Uygulama kontrollerinin etkinliğini değerlendirmek ve zayıflıklarını tespit etmek için test stratejileri oluşturulmalıdır. Testlerin belirli bir düzen içinde gerçekleştirilmesi, tüm kontrol unsurlarının doğruluğunu sağlamada etkili bir rol oynar. Test stratejisi belirlerken, her bir kontrol türünün kapsamlı bir şekilde test edilmesi ve denetlenmesi gerektiği göz önünde bulundurulmalıdır.

Denetçi, kontrol etkinliğini değerlendirebilmek için planlanan testlerin sonuçlarını analiz eder. Bu testler, kontrollerin ne kadar etkili olduğunu ve organizasyonel hedeflere ne kadar ulaşılabildiğini gösteren somut verilere dayanmalıdır. Ayrıca, kontrol zayıflıklarının etkilerini inceleyerek, bu zayıflıkların sistemin genel güvenliğine ve verimliliğine nasıl zarar verdiğini ortaya koymak mümkündür.

Uygulama Kontrollerinin Sürekli İzlenmesi: Kontrollerin etkinliğini ve işlevselliğini sağlamak, bir defaya mahsus yapılan işlemlerle sınırlı değildir. Sürekli izleme ve düzenli gözden geçirme, sistemin uzun vadeli sağlığını ve güvenliğini sağlamak için kritik öneme sahiptir. Uygulama kontrollerinin sürekli olarak izlenmesi, organizasyonların sistemdeki potansiyel zayıf noktaları hızla tespit etmelerini ve hızlıca düzeltici önlemler almalarını sağlar. Bu aynı zamanda, sistemin gelecekteki kullanımında karşılaşılan yeni tehditlere karşı daha dayanıklı hale gelmesine yardımcı olur.

Uygulama kontrolleri, yalnızca bir sistemin güvenliğini sağlamakla kalmaz, aynı zamanda iş süreçlerinin her aşamasında etkinliğin ve verimliliğin arttırılmasını da sağlar. Uygulama kontrollerinin kapsamlı bir şekilde uygulanması, organizasyonların hedeflerine ulaşmasını kolaylaştırırken, güvenlik ihlallerinin ve sistem hatalarının olasılığını da azaltır. Sonuç olarak, uygulama kontrolleri, organizasyonel sürdürülebilirliğin sağlanmasında ve yazılım projelerinin başarıya ulaşmasında önemli bir araçtır.

2.3.1. Giriş/Kaynak Kontrolleri

Girdi kontrolleri, yapılan işlemlerin yönetim tarafından onaylandığını ve yetkilendirmelerin yönetim tarafından yapıldığını doğrular. Sadece geçerli sayılan ve yetki verilen verilerin girilmesini ve bu işlemleri sadece bir kez yapılmasına olanak sağlar.

a. Yetkilendirmeler

Yetkilendirme süreci ile aşağıdakiler sağlanır:

- İlgili belgelerde yer alan imzaların yetkili kişiler tarafından atılıp atılmadığının kanıtı,

- Basılı/dijital belgelerdeki kaynakların referans kontrolleri,

- Erişim yetkilerinde kısıtlamaları, birbirinden farklı parola yönetimini ve olası yetkisiz erişim durumlarında raporlamanın yapılması,

- Sadece yetki verilen kişiler tarafından, erişim izni verilen verilere erişim ve işlemleri yapabilmelerine imkân sunması,

- Her iş alanlarında yetkilerin ve iş adımlarının çalışan rol tanımlarına göre sınırlandırılması.

Yetkilendirme süreci, bilgi sistemlerinin güvenliğini sağlamak ve yalnızca yetkili kişilerin hassas verilere erişmesini temin etmek için kritik bir öneme sahiptir. Bu süreçte atılacak her adım, sistemin bütünlüğünü, güvenliğini ve iş sürekliliğini korur. Aynı zamanda yetkilendirme, organizasyonun veri güvenliğini sağlamak ve kullanıcıların yalnızca gerekli olduğu verilere erişim izni almasını sağlamak amacıyla sıkı kontrol mekanizmaları oluşturulmasını gerektirir.

Erişim Düzeylerinin Belirlenmesi: Kullanıcıların hangi verilere erişebileceği, hangi işlemleri yapabileceği, sistemdeki her bir görev için tanımlanmış yetki seviyelerine dayanmalıdır. Erişim düzeylerinin belirlenmesi, sadece gerekli olan bilgilere sınırlı erişim sağlanarak gereksiz veri sızıntılarını önler.

Zaman Kısıtlamaları ve Geçici Yetkilendirme: Belirli bir süre için verilen erişim izinleri, bir kullanıcının belirli bir işlem veya görev için yalnızca gereksinim duyduğu süre boyunca verilmelidir.

Bu tür geçici yetkilendirme, yalnızca görev tamamlandığında iptal edilmelidir ve kullanıcıların gereksiz erişim haklarına sahip olmasını engeller.

Erişim İzleme ve Raporlama: Yetkilendirmelerin denetlenmesi, belirli bir süre boyunca kimlerin hangi verilere eriştiğini ve hangi işlemleri gerçekleştirdiğini takip etmek için erişim günlükleri tutmayı içerir. Bu, güvenlik ihlallerinin hızlıca tespit edilmesine olanak tanır ve aynı zamanda kötüye kullanım durumlarında geri izleme yapılmasını sağlar.

Çok Faktörlü Kimlik Doğrulama (MFA): Erişim güvenliğini artırmak için MFA gibi ek güvenlik önlemleri kullanılmalıdır. Bu yöntem, kullanıcıların sisteme giriş yapmadan önce kimliklerinin doğrulanmasını sağlar ve sadece şifrelerle yapılan girişlere kıyasla daha güvenli bir ortam sunar.

Yetkilendirme süreci, organizasyonların yalnızca belirli ve güvenilir kullanıcıların kritik verilere erişebilmesini sağlayarak, dış tehditlere karşı korunmasına yardımcı olur ve iç tehditleri minimize eder. Bu nedenle, bu sürecin doğru şekilde uygulanması ve izlenmesi gereklidir.

b. Toplu Kontroller ve Mutabakat

Toplu kontrollerin sağlanması için girdilerde işlem kontrolleri toplu yapılır ve kontroller sonucunda mutabakatlar sağlanır. Toplu kontroller ve mutabakat, bir organizasyonun finansal işlemleri ve kayıtlarını doğrulamak ve güvence altına almak için kullanılan önemli bir süreçtir. Bu süreç, işlemler sırasında hataların veya dolandırıcılığın erken tespit edilmesine yardımcı olur ve finansal verilerin doğruluğunu sağlar. Bu kapsamda kontrol edilen veriler:

- Toplam Para Miktarı: Organizasyonların finansal hesaplarını doğrulamak için toplam para miktarları kontrol edilir. Bu, işlemlerin ve hesapların doğruluğunu belirlemeye yardımcı olur.

- Dökümanların Toplamı: Finansal dökümanların toplamı, kayıtların eksiksiz ve tutarlı olduğundan emin olmayı amaçlar. Her belgenin doğruluğu ve eksiksizliği kontrol edilir.

- Toplama Ait Özetler: Toplamların ve özetlerin tutarlılığı ve doğruluğu kontrol edilir. Özellikle finansal raporların hazırlanmasında bu tür özetler önemlidir.

- Hesaplarda Kontroller: Organizasyonun finansal hesaplarının mutabakatı yapılır. Banka hesapları, kredi kartları ve diğer finansal hesaplar gözden geçirilir.

- İş Kayıtlarının Toplamı: İş kayıtları ve işlem geçmişleri, finansal kayıtların tutarlılığını ve doğruluğunu sağlamak için kontrol edilir.

- Anlaşmalar: Finansal anlaşmalar ve sözleşmeler, ilgili taraflar arasında mutabakat sağlamak için kontrol edilir.

- Oluşturulan Toplam Kalemler: Finansal hesaplamalar sırasında oluşturulan toplam kalemler ve hesaplamalar, doğruluğu ve tutarlılığı kontrol etmek için incelenir.

Toplu kontroller ve mutabakat, finansal yönetim ve raporlama süreçlerinin güvenilirliğini artırır ve finansal sahtekarlık veya hataların tespit edilmesine yardımcı olur. Ayrıca, organizasyonların mali tablolarını doğrulamak ve dış denetçilere veya düzenleyici kurumlara karşı uyumluluğunu kanıtlamak için önemlidir.

Toplu kontroller, özellikle finansal işlemlerin doğruluğunu sağlamak ve mali verilerin güvenliğini temin etmek açısından kritik bir öneme sahiptir. Bu süreç, yalnızca mevcut verilerin doğruluğunu sağlamakla kalmaz, aynı zamanda organizasyonun operasyonel verimliliğini ve finansal şeffaflığını artırmak için de kullanılır. Ancak günümüzde finansal yönetimdeki değişikliklerle birlikte, toplu kontrollerin nasıl işlediği, nasıl entegre edildiği ve hangi modern yaklaşımların bu süreçlere dahil olduğu önem kazanmıştır.

Dijitalleşmenin artan etkisiyle birlikte toplu kontrollerin yerini alan yeni sistemler, bu sürecin hızını ve doğruluğunu önemli ölçüde artırmıştır. Gelişmiş yazılım çözümleri, toplu verileri sadece hızlı bir şekilde işlemiyor, aynı zamanda verilerin doğruluğu ve güvenliği konusunda proaktif analizler de sunuyor. Örneğin, finansal verilerin toplu olarak işlenmesi, sadece bir kontrol noktası olmaktan çıkmış, gerçek zamanlı verilerin anlık kontrolü ile desteklenen bir sürece dönüşmüştür. Bu değişim, insan hatalarının minimize edilmesini ve sahtekarlıkların daha erken tespit edilmesini sağlamaktadır.

Yapay zeka (AI) ve makine öğrenimi (ML), toplu kontrollerin işleyişini otomatikleştirerek hem zaman hem de kaynak tasarrufu sağlamaktadır. AI tabanlı sistemler, büyük veri setleri üzerinde çalışarak, belirli finansal eğilimleri ve kalıpları tespit edebilir. Bu sayede, hatalı verilerin tespit edilmesi, normalden sapmaların fark edilmesi ve yanlış hesaplamaların önlenmesi hızlandırılabilir. Örneğin, yapay zeka destekli yazılımlar, organizasyonların geçmiş finansal verilerine dayalı olarak potansiyel hataları önceden tahmin edebilir, bu sayede her türlü hata daha sistematik bir şekilde tespit edilebilir.

Blockchain teknolojisi, finansal verilerin güvenli bir şekilde kaydedilmesini ve denetlenmesini sağlayarak toplu kontrollerin doğruluğunu artırmaktadır. Blockchain'in en büyük avantajı, verilerin değiştirilemez ve şeffaf bir şekilde saklanmasıdır. Bu, toplu kontrol süreçlerinin her aşamasının doğruluğunu güvence altına alırken, her türlü manipülasyonun tespit edilmesini kolaylaştırır. Özellikle, ödeme işlemleri ve finansal raporlama sistemlerinde blockchain kullanımı, verilerin doğru şekilde kaydedilmesi ve her bir işlemin izlenmesi için önemli bir araçtır.

Toplu kontrollerin en önemli işlevlerinden biri, doğru finansal raporların oluşturulmasını sağlamaktır. Her bir işlem ve ödeme kaydının doğruluğu, bir organizasyonun finansal tablosunun şeffaflığını artırır. Bu bağlamda, toplu kontroller sadece hata tespitine yönelik değil, aynı zamanda düzenleyici kurumların uyum gereksinimlerini yerine getirmek adına da önemli bir rol oynar. Finansal raporlar ve gelir tabloları, yasal gereksinimlerin yerine getirilmesinin yanı sıra, organizasyonun iş yapma biçimlerinin de güvence altına alınmasını sağlar.

Toplu kontrollerdeki veri mutabakatı süreci, finansal hesapların doğruluğunun sağlanmasında kritik bir aşamadır. Özellikle çok sayıda işlem yapılan büyük organizasyonlarda, mutabakatın sağlanması zor olabilir. Ancak, veri doğrulama süreçlerinin modern yazılım araçlarıyla entegre edilmesi, bu süreci daha verimli hale getirmiştir. Örneğin, bankacılık sektöründe kullanılan mutabakat yazılımları, tüm hesapları, kredi kartı ödemelerini ve borç ilişkilerini anlık olarak izler ve birbirleriyle karşılaştırır, bu sayede herhangi bir hata hızla tespit edilebilir. Ayrıca, dış denetçiler veya düzenleyici kurumlar tarafından yapılan denetimlerde bu verilerin sağlıklı ve şeffaf olması gerekmektedir.

Büyük veri ve ileri düzey analitik araçları, toplu kontrollerin etkinliğini artıran önemli bir diğer bileşendir. Toplu verilerin analiz edilmesi, sadece verilerin doğruluğunu sağlamakla kalmaz, aynı zamanda finansal geleceği tahmin etme, risk analizleri yapma ve stratejik kararlar almayı da kolaylaştırır. Veri görselleştirme teknikleri ve raporlama araçları, yöneticilere ve denetçilere daha net bir analiz sunarak, tüm organizasyonda verilerin daha etkili bir şekilde kullanılmasını sağlar.

c. Hata Raporlanması ve Aksiyon Takibi

Girdi hatası işleme, sisteme sadece doğru verinin kabul edilmesini doğrular. Bu kapsamda aşağıdaki işlemler yapılır:

- Hatalı işlemler ret edilir, paketin (batch) geri kalanında işleme sürecine devam edilir.

- Hata olsa bile işleme sürecine devam edilir. Hata olan girdilerde sonradan hata düzeltme yapılması için işaretleme sağlanır.

- Hata içeren tüm paket (batch) düzeltilmesi için reddedilir.

- Hata içeren parti, düzeltilene kadar askıya alınır ve şüpheli işlem olarak bekletilir. Ret edilme işlemi uygulanmaz.

Hata raporlanması ve aksiyon takibi konusundaki açıklamalara bir örnek olarak:

Bir e-ticaret şirketi, müşteri siparişlerini işlemek için bir otomasyon sistemi kullanmaktadır. Bu sistem, müşterilerin siparişlerini alır ve ödeme işlemlerini yapar. Ancak bazen sistemde hatalı veri girişi nedeniyle siparişlerde sorunlar oluşabilir. Örneğin, bir müşteri yanlışlıkla 20 adet ürün siparişi verdiğinde, sipariş sistemi bu siparişi doğru bir şekilde işleme almamalıdır, çünkü bu sipariş anormal bir büyüklüğe sahiptir. İşte bu noktada hata raporlanması ve aksiyon takibi devreye girer:

- Hatalı işlem ret edilir: Sistem, abartılı bir siparişi otomatik olarak tanır ve işlemi reddeder. Bu, normal işlem sırasını bozmadan hata olan siparişi ayırır.

- Hata olsa bile işleme sürecine devam edilir: Hata raporlandığında, sistem siparişi işlemeye devam eder, ancak bu hata kaydedilir.

- Hata düzeltme işaretleme sağlanır: Hata, siparişin ayrıntılarında işaretlenir ve sistemdeki çalışanlar tarafından fark edilir. Bu, hata düzeltilene kadar siparişi takip etmeleri gerektiği anlamına gelir.

- Hata içeren parti düzeltilene kadar askıya alınır: Sistem, bu hatalı siparişi otomatik olarak bir "şüpheli siparişler" kuyruğuna yerleştirir. Siparişin neden hatalı olduğu ve nasıl düzeltileceği incelenir. Eğer sorun düzeltilmezse, sipariş işleme alınmaz.

Bu şekilde, hatalı siparişlerin işlenmesi sırasında doğru verilerin sisteme girmesi ve hataların düzeltilmesi sağlanır. Bu, müşteri memnuniyetini artırır ve işletme verimliliğini korur.

Girdi hatası işleme, herhangi bir hata oluştuğunda sistemin doğru veri kabul etmesini sağlamak için kritik bir süreçtir. Bu süreç, özellikle büyük veri ve otomasyon sistemlerinin hakim olduğu çağımızda, işletmelerin verimliliğini artırırken, hata oranlarını minimize etme konusunda da önemli rol oynamaktadır. Ancak günümüzün hızlı değişen iş ortamında, hata raporlaması ve aksiyon takibi de artık yalnızca manuel kontrol ve takipten ibaret değildir. Modern teknolojiler ve otomasyon araçları, bu süreçleri çok daha verimli, hızlı ve doğru bir şekilde yönetilmesini sağlar.

Yapay Zeka Destekli Hata Tespiti ve Otomatik Düzenlemeler

Yapay zeka (AI) ve makine öğrenimi (ML) gibi teknolojiler, hata raporlama ve aksiyon takibi süreçlerini daha proaktif hale getirmektedir. Özellikle e-ticaret ve finans sektöründe, otomasyon sistemleri, verileri anında kontrol ederek herhangi bir yanlışlık durumunda sistemi otomatik olarak uyarabilir. AI destekli hata tespiti araçları, yalnızca hatalı verileri tespit etmekle kalmaz, aynı zamanda bu hataların gelecekteki tekrarlamalarını engellemek için önerilerde bulunabilir. Örneğin, bir müşteri siparişi sırasında 20 adet yerine 200 adet sipariş verilmesi durumunda, bu tür anormallikleri AI destekli sistemler anında fark eder ve işlem yapılmadan önce kullanıcıyı uyarır.

İleri Düzey Analitik ve Hata Kaydı İzleme

Büyük veri ve analitik araçları, hata raporlama sürecinin daha hızlı ve etkili bir şekilde yönetilmesine yardımcı olur. Veri görselleştirme teknikleri, büyük işlem kümelerindeki (batch) hataların daha hızlı tespit edilmesini sağlar. Örneğin, çok sayıda sipariş veri tabanı üzerinden yapılan analitikler, işlem sırasında oluşan hataların hemen raporlanmasını ve ilgili işlemin izlenmesini sağlar. Bu tür verilerle yapılan analizler, hataların nedenlerini hızlıca belirlemeyi ve çözüm önerileri geliştirmeyi kolaylaştırır. Bu sayede, daha doğru ve kapsamlı hata raporları oluşturulabilir.

Blockchain ve Hata Raporlama: Güvenilirlik ve Şeffaflık

Blockchain teknolojisi, veri güvenliği ve şeffaflık açısından devrim yaratmaktadır. Hata raporlama ve aksiyon takibi sürecinde de blockchain, verilerin doğruluğunu ve güvenliğini artırmak için kullanılabilir. Herhangi bir hatalı işlem, blockchain üzerinde kaydedildiğinde, geri dönülmesi ve düzeltme yapılması için daha şeffaf bir süreç ortaya çıkar. Bu sayede, tüm paydaşlar hatalı işlemlerin nasıl düzeltildiğini ve sürecin nasıl yönetildiğini net bir şekilde takip edebilir. Blockchain, aynı zamanda verilerin geri dönülemezliğini sağlar ve yapılan her değişikliğin kaydını tutarak, gelecekteki denetim süreçlerini kolaylaştırır.

Regresyon Testleri ve Etkili Hata Yönetimi

Hataların düzeltilmesinin ardından yapılan regresyon testleri, sistemin geri kalan bölümlerinin etkilenip etkilenmediğini kontrol etmek için kritik bir öneme sahiptir. Bu testler, yazılım geliştirme süreçlerinde özellikle yazılımın belirli bir güncellemeden sonra beklenmedik şekilde çalışmaya devam etmesini sağlamak için kullanılır. Hata yönetimi sürecinde, herhangi bir düzeltmenin ardından yapılacak regresyon testleri, tüm sistemin bütünlüğünü koruyarak işletme faaliyetlerinin sorunsuz bir şekilde devam etmesini sağlar. Bu, sistemin güvenliğini ve verimliliğini artırırken, potansiyel hataların önceden tespit edilmesini sağlar.

Proaktif Bakım ve Sürekli İyileştirme

Hata raporlama ve aksiyon takibi, sadece hataların düzeltilmesiyle sınırlı kalmamalıdır. Bu süreçler, sürekli bir iyileştirme döngüsüne dönüşmelidir. Proaktif bakım yöntemleri ve sürekli izleme araçları, sistemdeki zayıf noktaları erkenden tespit ederek iyileştirme fırsatları sunar. Bu sayede, hataların olası etkilerini minimize etmek için önceden aksiyon alınabilir. Özellikle büyük verilerin işlendiği ortamlarda, sistemin her zaman izlenmesi ve sürekli olarak iyileştirilmesi gerektiği unutulmamalıdır.

2.3.2. İşleme Prosedürleri ve Kontrolleri

Uygulama işlemlerinin güvenirliğini sağlamak için işleme prosedür ve kontrolleri takip edilir. İşlem kontrolleri, girilen veriler ile istenen çıktıları üretecek şekilde işlenmesini sağlar. Tüm işlemlerin, güvenlik ilkeleri çerçevesinde gerçekleştirilmesine dikkat edilmelidir.

İşleme prosedürleri ve kontrolleri, sistemlerin güvenli ve etkin çalışabilmesi için kritik öneme sahiptir. Bu kontrollerin etkin bir şekilde uygulanması, doğru veri işleme, güvenlik ihlallerinin önlenmesi ve sistemin kesintisiz çalışmasını garanti eder. İşleme prosedürlerinin başarısı, verilerin doğru şekilde işlendiği, güvenliğin sağlandığı ve organizasyonel hedeflere ulaşılabildiği ölçütlere dayanır. İşlem süreçlerinin takibi ve denetimi için özellikle dikkat edilmesi gereken noktalar vardır.

Girilen verilerin doğruluğu, doğru bir işlem çıktısı elde etmek için temel bir gerekliliktir. Verilerin doğruluğu, yalnızca ilk girişte sağlanmakla kalmaz, aynı zamanda veri işleme ve hesaplama süreçlerinde de sürekli kontrol edilmelidir. Herhangi bir hata, ilerleyen süreçlerde daha büyük sistem hatalarına yol açabilir. Bu yüzden, veri doğruluğu ve tutarlılığı, işlem süreçlerinin temel kontrol unsurlarından biridir. Ayrıca, uygulama işlemlerinin her adımının izlenmesi gerekmektedir. Bu izleme, işlem sırasında oluşan her türlü hata, eksiklik veya anormalliğin tespit edilmesini sağlar. Özellikle büyük ve karmaşık sistemlerde, her işlemin kaydının tutulması ve izlenmesi, şeffaflık ve denetlenebilirlik sağlar. Otomatik izleme araçları ve log tutma sistemleri, işlemlerin doğru bir şekilde gerçekleştirilip gerçekleştirilmediğini denetler.

İşlem kontrolleri, yalnızca yetkili kullanıcıların işlem yapabilmesini sağlar. Her kullanıcıya, rolüne ve görevine uygun erişim hakları verilmelidir. Yetkisiz erişimlerin önlenmesi için sıkı güvenlik önlemleri alınmalı, tüm işlemler kayıt altına alınmalıdır. Bu, sistemin hem güvenliğini sağlar hem de işlem hatalarına karşı koruma sunar. Sistemde her işlem için belirli protokoller belirlenmeli ve istisnalarla karşılaşıldığında nasıl bir yaklaşım izleneceği önceden tanımlanmalıdır. İstisnalar, genellikle beklenmeyen durumlarla ilgilidir ve sistemin normal işleyişinin dışında kalan anormal koşulları içerir. İstisnalarla karşılaşıldığında, işlemin durdurulması veya geri alınması gibi prosedürler devreye girmelidir. İşlem sonrası elde edilen çıktılar düzenli olarak izlenmeli ve raporlanmalıdır. Herhangi bir hata veya tutarsızlık meydana geldiğinde, bunlar hemen ilgili birimlere bildirilerek düzeltilmelidir. Raporlama sistemlerinin doğruluğu, organizasyonel şeffaflık için büyük önem taşır.

Ayrıca, herhangi bir işlem hatası durumunda işlemin geri alınabilmesi veya düzeltilmesi için bir kurtarma prosedürü bulunmalıdır. Veri kaybı, sistem hataları veya yanlış işlem sonuçları gibi durumlar için yedekleme ve geri yükleme prosedürleri oluşturulmalıdır. Bu, iş sürekliliğini sağlamak ve sistemin tekrar eski haline gelmesini sağlamak için kritik öneme sahiptir.

Sonuç olarak, işleme prosedürleri ve kontrolleri, hem güvenlik hem de veri doğruluğunu sağlamak için dikkatlice tasarlanmalı ve izlenmelidir. Etkin işlem kontrolleri, tüm süreçlerin güvenli bir şekilde ilerlemesini sağlar, hataların önlenmesine yardımcı olur ve sistemin sorunsuz çalışmasını garanti eder. Teknolojinin ilerlemesiyle birlikte bu kontrollerin etkinliği artırılmalı ve dijitalleşme sürecinde güncel kontrol yöntemleri entegre edilmelidir. Bu şekilde, organizasyonların hedeflerine ulaşmasını sağlayacak güçlü ve güvenli bir işlem altyapısı kurulmuş olur.

a. Veri Doğrulama ve Düzenleme Prosedürleri

Sisteme girilecek verilerin hatasız olması için, veri doğrulama ve düzeltme prosedürleri kullanılır. Bu prosedürler ile eksik, hatalı ya da ilgisiz verilerin tespit edilmesi sağlanır. Takip edilen prosedürlere yönetici tarafından manuel müdahaleler varsa, bu müdahalelere ilişkin kayıtların tutulması ve periyodik olarak gözden geçirilmesi sağlanmalıdır. Veri doğrulama ve düzenleme prosedürlerine aşağıda yer verilmektedir.

- Sıra kontrolü: İşlemlere ait kontroller bir sırayı takip etmelidir. Tekrar eden numara ya da beklenmeyen kontrol numaraları ile karşılaşılması durumunda ret edilir ya da istisna olarak kayıt altına alınır. Örneğin oluşan faturalar sıra numarasına göre kayıt altına alınır. İlgili günün ilk faturası belirli bir sayı ile başlar ve yine aynı şekilde belirli bir numara ile bitmesi beklenir. Eğer arada farklı numaralar ile karşılaşılırsa, faturaya ait hatalı girdi olarak kabul edilir.

- Limit kontrolü: Veriler belirlenmiş bir miktara kadar girilebilecektir. Örneğin maaş verileri X TL'yi geçmeyecektir. Bu rakamın üzerinde bir maaş verisi girildiğinde sistem tarafından reddedilecektir.

- Aralık kontrolü: Veriler belirli bir değer aralığında olmalıdır. Maaş örneğinden gidilirse, en düşük maaş ile en yüksek maaş arası olmalıdır şeklinde bir tanımlama yapılabilir. Bu durumda belirlenen değerlerin dışı geçersiz kabul edilecektir.

- Geçerlilik kontrolü: Daha önceden belirlenen kriterlere uygun formatta veri girişi beklenmektedir. Örneğin maaş bilgisi alanında medeni durum ile ilgili veri girişi yapılacak alana sadece E veya B harfleriyle giriş imkânı tanınır. Başka bir sembol, rakam harf girişi yapılamaz.

- Makullük kontrolü: Veriler önceden belirlenen, makul değerler sınırı içerisinde kalmalıdır. Bunun dışında veri girişi yapılması durumunda hata mesajı gelecektir. Maaş örneğinden gidecek olursak, veri girilecek alana milyon gibi bir rakam girildiğinde sistem hata verecektir veya ilgili alana, beklenenden çok farklı bir rakam girilmesi ile ilgili hata mesajının oluşmasına neden olacaktır.

- Tablo bakma listeleri: Önceden tanımlanan kriterlere ait liste ile girdi olarak sağlanan verilerin uyumlu olması gerekmektedir. Listede yer almayan değerlere ait veri girişi yapılması engellenir.

- Mevcudiyet kontrolü: Önceden belirlenen kriterlere göre veri girişleri sağlanır. Örnek olarak; doğum tarihi, telefon numarası vb. verilebilir.

- Anahtar doğrulaması: Tuş vuruşları ile doğru kişi tarafından verinin giriş yapıp yapmadığı doğrulanır. En farklı kontrol yöntemlerinden birisidir.

- Kontrol basamağı: Matematiksel olarak bir değer hesaplanır ve aktarılacak verilere eklenir. Aktarım sırasında verinin bütünlüğünün bozulmasına yönelik bir hata olup olmadığı anlaşılır.

- Bütünlük kontrolü: Veri giriş alanlarında her zaman 0 ya da boşluktan farklı girişlerin yapılması beklenir. Örneğin, işe yeni giren bir personelin T.C. kimlik no alanının dolu olması gerekmektedir. Boş olması durumunda giriş ret edilir.

- Mükerrerlik kontrolü: Geçmişte yapılan girişler ile yeni yapılan girişler karşılaştırılarak mükerrer giriş olması engellenir. Örneğin, tedarikçiye ait fatura numarası ile önceden ödemesi tamamlanan faturalar kontrol edilir ve iki kez ödeme yapılmasının önüne geçilir.

- Mantıksal ilişki kontrolü: Bir ya da daha fazla koşulu sağlayan veriler doğru kabul edilir. Örneğin, tam zamanlı çalışma kapsamında çalışan profilinde 25 yaş sınırı varsa, işe alınma tarihinin doğum tarihinden en az 25 yıl geçmiş olması beklenir.

b. İşleme Kontrolleri

Biriken verilerin doğruluğunu ve tamlığını doğrulamak için kullanılır. Yetki kapsamında gerçekleşen işlemler dışında veri tabanında bulunan verilerin doğru ve eksiksiz kalması sağlanır. İşleme kontrollerine ana başlıkları itibarıyla aşağıda yer verilmektedir.

- Manuel yeniden hesaplama: Beklenen görevi yerine getirdiğinin doğrulanması adımıdır.

- Düzenleme: Veri girişlerinde, verilerin doğruluğu ve tamlığı kontrol edilerek hata tespit edilirse, otomasyon aracı veriyi giren kişiye hatayı bildirir ve işlem kabul edilmez.

- Çalıştırmadan çalıştırmaya kontrol toplamları: İşleme süreci boyunca veri değerlerinin verifikasyonudur. Okunan veriler süreç içerisinde güncellenir.

- Programlanmış kontroller: Veri ve işlemdeki hataları tespit eden ve düzeltici faaliyetleri otomatik olarak başlatan programlardır. Örneğin, hatalı veriler işlenmek üzere girilmişse, uygulama program uygun verilerin kullanılması gerektiğini bildiren iletileri görüntüleyebilir.

- Hesaplanan tutarların makul olarak doğrulanması: Miktarlarda hesaplama sonucunda uygunluğu için doğrulama yapar. Uygunluk önceden belirlenen kriterlere göre yapılmaktadır. Uygun bulunmayan işlemler incelenmek üzere ret edilir.

- Miktarlar üzerindeki limit kontrolleri: Daha önceden belirlenen limitler ile tutarların doğruluğu kontrol edilir. Limitler dışında olan işlemler daha sonra incelenmek üzere ret edilir.

- Dosya toplamları mutabakatı: Mutabakat, geçmişe ait manuel tutulan hesaplara ilişkin kontrol kayıtları ya da bağımsız kontrol dosyaları kullanılarak yapılır.

- İstisna raporları: İstisna raporları önceden belirlenen kriterler dışında yer alan yanlış işlemler veya verilerle ilgili bilgi sistemi tarafından oluşturulur.

c. Veri Dosyası Kontrol Prosedürleri

Veri dosyası kontrol prosedürleri, depolanan veriler üzerinde sadece yetkili kişiler tarafından işlem yapılabilmesine olanak sağlar. Bu kontrol prosedürlerine aşağıda yer verilmektedir.

- Düzeltme önce ve sonrası imaj raporlaması: Dosyada yer alan veriler için ilgili işlem öncesi ve sonrası imajları alınır. Böylece gerçekleşen işlemlerin veriler üzerindeki etkisi anlaşılmaya çalışılır.

- Bakım hatası raporlama ve işleme: Tüm hata raporlarına ilişkin gerekli aksiyonların ve düzeltmelerin zamanlı olarak yapıldığının teyit edilmesi için kullanılır. Düzeltme işlemi yapacak kişi ile düzeltme işleminin doğruluğunu teyit eden kişilerin farklı olması, görevler ayrılığı ilkesi için kritiktir.

- Kaynak belge saklama: Kaynak dokümanlar, işleme tabi tutulan verilerin geri getirilmesi, düzeltilmesi, teyit edilmesi için belirli bir süre tutulmalıdır. Bu saklanan verilere sadece yetkili kişiler iş ihtiyaçları çerçevesine erişebilmelidir. Bu belgelerin imha edilme süreci de güvenli ve kontrollü bir şekilde gerçekleştirilmelidir.

- İç ve dış etiketleme: Çıkarılabilir depolama araçları/medyaları için iç ve dış etiketleme yapılmalıdır. Bu yöntem ile uygun verinin işlem için kullanıldığı teyit edilir. Dış etiketleme ile doğru medya aracının kullanıldığı, iç etiketleme ile de bu medyadaki ilgili veri dosyalarının kullanıldığı teyit edilmiş olur.

- Versiyon kullanımı: Verilerin en güncel versiyonlarının veya istenilen versiyonlarının kullanılmasını sağlar. Örneğin, işlemler verilerin en güncel versiyonu ile yapılır ancak geri döndürme adımları verilerin eski versiyonu üzerinden gerçekleştirilir.

- Veri dosyası güvenliği: Yetkisiz kişiler tarafından yapılacak erişimler engellenir, ancak yetkili kişiler tarafından yapılan hatalı işlemlerin engellenmesi sağlanamaz.

- Bire-bir kontrol: İşlenmesi beklenen doküman listesi ile bilgi sistemleri tarafından işlenen döküman listeleri karşılaştırılır. Dosyaların istenilen şekilde işlendiğinin teyit edilmesi sağlanır.

- Önceden kaydedilmiş giriş: Veri girişlerinde hataların önlenebilmesi için kullanıcı girmeden önce bazı verilerin girilmiş olarak gelmesi sağlanabilir. Örneğin, kullanıcı adı ve şifresinin hazır gelmesi gibi.

- İşlem hareket günlükleri (loglar): Büyük işlemler bilgi sistemleri tarafından kayıt altına alınır. Denetim izi sağlaması için, giriş alanı, işlemin tarihi, kullanıcı adı ve terminal alanı gibi bilgiler tutulur. Böylece, ilgili kişiler tarafından hangi işlemlere ait kayıtların tutulacağının veya raporlanacağının belirlenmesine ve sistemsel hataların tespit edilmesini sağlar.

- Dosya güncelleme ve erişim yetkileri: Önleyici kontrol olarak depolanan verinin uygun şekilde korunmasını ve güncelliğini sağlar. Genel giriş yetkilerin dışında kalan bazı uygulama girişler için de sınırlamalar getirilebilir. Sadece yetkili kişilerin ilgili alanlara girişi sağlanmalıdır. Böylece log kayıtlarının alınması yanında güçlü kontrol ortamı da sağlar.

Parite kontrolü (eşlik sınaması): Aktarılan veride meydana gelen tek sayıda hataları sezmek için kullanılır. Böylece, verideki birlerin sayısını tek ya da çift olacak şekilde düzenlemektir. Veriye bir eşlik biti eklenir. Eşlik biti bir ya da sıfır yapılarak tüm veri grubu içindeki birlerin sayısının çift ya da tek olması sağlanır. Birlerin sayısının çift olmasına çift eşlik, tek olmasına da tek eşlik durumu denir.

2.3.3. Çıktı Kontrolleri

Çıktı kontrolleri, kullanıcılar için oluşturulan verilerin tutarlı ve güvenli sunulabilmesini ve iletilmesini sağlar. Bu kontrollere aşağıda yer verilmektedir.

- Hassas ve kritik verilerin güvenli bir yere kaydedilmesi ve depolanması: Kritik verileri içeren formların olası problemler nedeniyle zarar görmesi ya da çalınması durumunda sürekliliği sağlamak adına güvenli bir alanda depolanması adımıdır. Depolanan veriler ile mevcut envanterler ile düzenli periyotlarda karşılaştırılmalıdır. Ortaya çıkan farklar ile ilgili araştırma yapılmalıdır.

- Bilgisayar üretimli devredilebilir araçlar, formlar ve imzalar: Üretilen tüm formlar ile eldeki listeler karşılaştırılmalıdır. Ortaya çıkan hatalar ve eksiklikler araştırılmalıdır.

- Rapor doğruluğu, tamlığı ve zamanlı olması: Güvenli olarak bilinen sistemlerden bile hatalı rapor çıktıları oluşabilmektedir. Bunu engellemek için rapor çıktılarını hazırlamaya yönelik hazır formatlar, şablonlar, değişiklik yönetimi talep rapor tasarımı ve oluşturma özellikleri, şablonlar hazırlanabilir.

- Sistemden üretilen raporlar: Yönetim tarafından iş kararları için kullanacağı veya iş süreçleriyle ilgili sonuçlar hakkında bilgiler içeren raporları tanımlamaktadır. Bu raporlarda yer alan verilerin doğru ve eksiksizliği bilgi sistemleri tarafından yapılması beklenen fonksiyondur. Sistemden elde edilen raporların doğruluğunun teyit edilmesi için aşağıdaki yöntemler kullanılabilir:

a. Rapor dağıtımı: Uygun ve onaylanmış dağıtım parametrelerine uyumlu çıktı raporları dağıtılmalıdır. İşlem adımları birimlere ait raporların tam ve zamanında iletmiş olduğunu teyit etmelidir. Rapor dağıtımları loglanmalı, dağıtım kanallarına erişim kısıtlanmalıdır. Hassas veri içeren raporlarda gizlilik ve erişebilirlik kontrolleri için ilave kontroller uygulanmalıdır.

b. Denkleştirme ve uzlaşma: Rapor çıktısı oluşmasını sağlayan uygulamaların oluşturduğu çıktılarda rakamsal değerler ve içeriğindeki bilgiler belirli periyotlarda kontrol edilmelidir. Denetim izleri üzerinden işlem adımları takip edilmelidir.

c. Çıktı hatası işleme: Gerçekleşen hatalara ilişkin rapor ve kontrol etme prosedürlerinin oluşturulması gerekmektedir. Zaman damgalı hata raporları olmalı, hataların düzeltilmesi ve incelemelerin sağlanması için işlem sahibine iletilmelidir.

d. Çıktı raporu saklama: Yasal düzenlemelerde yer alan kayıt saklama ile ilgili yönergeler politikalara dahil edilmelidir. Bu kapsamda hukuk birimlerinden görüş ve yönlendirmeleri kritik önem taşır.

e. Alındı raporları doğrulaması: Kritik verilere ait oluşturulan raporların ilgili kişi ya da kişiler tarafından alındığına ilişkin log kayıtlarının, çıktıların dağıtım prosedürüne uygun olarak yapıldığına dair kanıt olarak saklanmalıdır.

Yazılım geliştirme sürecinde çıktı kontrollerinin neden önemli olduğunu anlamak için bir yazılım örneği düşünelim:

Senaryo: Yazılım Sürüm Yayını

Bir yazılım geliştirme şirketi, yeni bir sürümünü kullanıcılara sunmayı planlamaktadır. Bu yeni sürüm, bir dizi yeni özellik ve hata düzeltmeleri içermektedir. Ancak yazılımın sürümünün yayınlanmadan önce, çıktı kontrollerinin titiz bir şekilde yapılması gerekmektedir.

Kod Denetimi: Yazılımın kaynak kodu, bir kod denetleyici veya otomasyon aracı tarafından kontrol edilmelidir. Bu, kodun hatalı veya güvenlik açıklarına sahip olup olmadığını belirlemeye yardımcı olur.

Birim Testleri: Yazılımın her bir bileşeni, birim testleri kullanılarak ayrı ayrı test edilmelidir. Bu testler, her bir bileşenin beklenen sonuçları üretip üretmediğini kontrol eder.

Entegrasyon Testleri: Yazılımın farklı bileşenlerinin bir araya geldiği yerlerde entegrasyon testleri yapılmalıdır. Bu, bileşenler arasındaki etkileşimleri ve veri akışını kontrol eder.

Sistem Testleri: Yazılımın tamamlanmış hali, tüm işlevselliğiyle test edilmelidir. Bu testler, yazılımın kullanım senaryolarını ve kullanıcıların işlemlerini simüle eder.

Performans Testleri: Yazılımın performansı, yük altında nasıl davrandığını belirlemek için test edilmelidir. Bu, kullanıcı sayısının arttığı durumları simüle eder.

Güvenlik Kontrolleri: Yazılımın güvenliği, potansiyel güvenlik açıkları ve zayıf noktalar açısından değerlendirilmelidir.

Belge Kontrolleri: Kullanıcılar için belgeler ve rehberler, doğru ve güncel olduğundan emin olunmalıdır.

Dağıtım Kontrolleri: Yazılımın dağıtımı sırasında, hatalı veya eksik dosyaların oluşmasını önlemek için kontroller yapılmalıdır.

Bu kontroller, yazılımın güvenilirliğini ve kalitesini sağlama sürecinin kritik bir parçasıdır. Eğer çıktı kontrolleri yeterince yapılmazsa, hatalı bir yazılımın kullanıcılara sunulması ciddi sonuçlara yol açabilir. Bu nedenle yazılım geliştirme sürecinde çıktı kontrolleri titizlikle uygulanmalıdır.

Çıktı Kontrollerinin Yazılım Geliştirme Sürecindeki Önemi

Çıktı kontrolleri, yazılım geliştirme sürecinde kullanıcıların güvenle kullanabilmesi için hatasız ve güvenli sonuçlar elde etmek adına kritik bir öneme sahiptir. Yazılımın doğru ve zamanında teslim edilmesi için sadece geliştirme sürecinde değil, aynı zamanda dağıtım öncesi çıktıların da titizlikle denetlenmesi gerekmektedir. Bu doğrultuda, yazılım sürümünün yayınlanmasından önce, çıktı kontrolleri yazılımın her aşamasında yapılmalıdır.

İlk adım olarak, yazılımın kaynak kodu üzerinde yapılan kod denetimi, yazılımın güvenliği ve kalitesi açısından büyük önem taşır. Bu denetim, kodun hatalı bölümlerinin ve potansiyel güvenlik açıklarının tespit edilmesini sağlar. Ayrıca, birim testleri, her bir bileşenin beklenen sonuçları verip vermediğini kontrol ederek, yazılımın farklı modüllerinin doğru çalıştığından emin olmayı sağlar.

Bileşenlerin birbirleriyle uyumlu bir şekilde çalışmasını sağlamak için entegrasyon testleri yapılmalıdır. Bu testler, yazılımın farklı bileşenlerinin birlikte çalışıp çalışmadığını ve veri akışının doğru şekilde gerçekleşip gerçekleşmediğini doğrular. Bunun ardından sistem testleri yapılır, bu testlerde yazılımın tüm işlevselliği kontrol edilir ve kullanıcıların yazılımla gerçekleştireceği işlemler simüle edilir.

Performans testleri, yazılımın yüksek yük altındaki davranışını test eder. Bu aşama, yazılımın birçok kullanıcının aynı anda erişim sağladığı durumlarda ne kadar verimli çalıştığını ve olası performans darboğazlarını belirler. Ayrıca, yazılımın güvenlik kontrolleri yapılmalı, potansiyel güvenlik açıkları ve zayıf noktalar tespit edilmelidir. Bu süreçte, şifreleme yöntemlerinin ve erişim izinlerinin düzgün çalıştığı doğrulanmalıdır.

Belge kontrolleri de yazılım geliştirme sürecinin bir parçasıdır. Kullanıcılar için hazırlanan belgeler ve rehberler, yazılımın doğru ve güncel sürümüne uygun olmalı, ayrıca kullanıcılara doğru bilgiler sunmalıdır. Dağıtım kontrolleri ise, yazılımın dağıtımı sırasında dosya bozulmalarını ve eksik dosyaların oluşmasını engellemeyi amaçlar. Bu kontrol adımları, yazılımın eksiksiz ve hatasız olarak kullanıcılara ulaşmasını sağlar.

Tüm bu kontrollerin yazılım geliştirme sürecine dahil edilmesi, kullanıcıların güvenli ve verimli bir yazılım deneyimi yaşamalarını sağlar. Eğer bu kontroller eksik veya yetersiz yapılırsa, yazılımın kullanıcılar üzerinde olumsuz etkiler yaratması kaçınılmaz olacaktır. Bu nedenle, çıktı kontrolleri, yazılımın güvenilirliğini artırmak ve kullanıcı memnuniyetini sağlamak adına temel bir süreçtir.

Örnek Sorular

Soru 1: Aşağıdakilerden hangisi "Sistem Geliştirme Yaşam Döngüsü" temel fazlarından biri değildir?

A) Değerlendirme

B) Geliştirme

C) Konfigürasyon

D) Tasarım

E) Fizibilite Çalışması

Cevap: A

Soru 2: Kritik başarı faktörleri aşağıdakilerden hangisini içermez?

A) İş verimi

B) Kalite

C) Ekonomik değer

D) Müşteri hizmeti

E) Tutarsızlık

Cevap: E

Soru 3: Uygulama kontrolleri ile ne sağlanması beklenmez?

A) İşlem sonuçlarının beklenti ile uyumunun kontrol edilmesi

B) Fizibilite çalışması

C) İşlemin doğru görevi tamamlayıp tamamlamadığının kontrol edilmesi

D) Verinin korunması

E) Doğru verileri kullanılması ve güncellemelerin yapılmış olmasının kontrol edilmesi

Cevap: B

Soru 4: Aşağıdakilerden hangisi üretim ortamına aktarım sırasında izlenen adımlardan olan "uygulamaya almanın planlanması" aşamasında yapılır?

A) Eğitim verilmesi

B) Fazlı geçiş planının oluşturulması

C) Canlıya geçişin sağlanması

D) Geri dönüş senaryosu gerçekleştirilmesi

E) Kullanıcı kabul testlerinin yapılması

Cevap: D

I Sistem güvenliğinin tasarım ile uyumluluğunun doğrulanması

II Kullanıcı Kabul testlerini kontrol ederek, Kabul edilen uygulamanın teslim edildiğinin doğrulanması

III Hata raporlarının kontrol edilmesi ve takibi için kullanılan prosedürlerin uyumluluğunun doğrulanması

IV Dahili kontroller için testlerin planlanıp gerçekleştiğinin doğrulanması

V son kullanıcılar ile görüşme yaparak, prosedürlerin, kullanım talimatlarının ve yeni yöntemlerin anlaşılırlığının doğrulanması

Soru 5: Yukarıda yer verilenden hangileri yazılım testlerinin kontrolü esnasında bilgi sistemleri denetçisinin dikkat etmesi gereken hususlardandır?

A) I, II, III

B) III, IV

C) III, V

D) II, III, V

E) Hepsi

Cevap: E

KAYNAKÇA

Kurumsal Çerçeve, Kütüphane, Standart, Yönetmelik ve Tebliğler:

COBIT 2019, Governance Management Objectives, ISACA, 2018.

ITIL kütüphanesi v3F

ISO Standartları

SPK Bilgi Sistemleri Yönetimi Tebliğ Madde 4 Tanımlar

SPK Bilgi Sistemleri Yönetimi Tebliği Madde 10

Kitap:

ISACA, "CISA Gözden Geçirme Klavuzu 27. Baskı", ISBN 978-1-60420-855-9

Information Reference Model: Quality Criteria for Information, COBIT 2019, Introduction and Methodology, 2018.

APO14.06 Ensure a data quality assessment approach, COBIT 2019, Governance and Management Objectives, 2018.

EBA Guidelines on ICT and security risk Management, 28 Kasım 2019, European Banking Authority.

Kazan, Halim. Proje Yönetimi, Ortak Ders. İstanbul Üniversitesi.

The NIST Definition of Cloud Computing (NIST Special Publication 800-145).

Cloud Security Alliance, "Best Practices for Mitigating Risks in Virtualized Environments.", 2015.

Web:

ISACA, https://www.isaca.org

PMI, http://projebilgialani.pmi.org.tr/

IU, https://ww4.ticaret.edu.tr/sbe/wp-content/uploads/sites/129/2020/03/projeyonetimiau207-%C4%B0stanbul%C3%9Cniversitesi.pdf

IU, http://auzefkitap.istanbul.edu.tr/kitap/endustrimuhlt_ue/projeyonetimi.pdf, Proje Yönetimi

IU, http://auzefkitap.istanbul.edu.tr/kitap/kok/projeyonetimiau207.pdf, Proje Yönetimi

Cumhuriyet Üniversitesi, http://pdo.cumhuriyet.edu.tr/wp-content/uploads/PROJE-Y%C3%96NET%C4%B0M%C4%B0..pdf

DergiPark, https://dergipark.org.tr/tr/download/article-file/560415

DergiPark, https://dergipark.org.tr/tr/download/article-file/277396

DergiPark, 328845 (dergipark.org.tr)

IMedium, https://medium.com/architectural-patterns/yazılım-geliştirme-modelleri-62915545c51e, YGM Modelleri

CSA, https://cloudsecurityalliance.org/blog/2021/04/06/what-an-auditor-should-know-about-cloud-computing-part-1/

CSA, https://cloudsecurityalliance.org/blog/2021/04/27/what-an-auditor-should-know-about-cloud-computing-part-3/

CSA, https://cloudsecurityalliance.org/research/working-groups/internet-of-things/

CSA, https://cloudsecurityalliance.org/research/working-groups/blockchain/

Bilişim. Konferans Bildirisi, https://ab.org.tr/ab07/kitap/salahli_yasar_AB07.pdf

EMO, Microsoft Word - 16.doc (emo.org.tr)

Tez, https://webdosya.csb.gov.tr/db/cbs/icerikler/salihsoylu_tez_v10-20180925134450.pdf

Tez, T09306.pdf (sakarya.edu.tr)

Wiki, Waterfall model - Vikipedi (wikipedia.org)

YTU, http://dspace.yildiz.edu.tr/xmlui/bitstream/handle/1/6206/0152763.pdf?sequence=1&isAllowed=y,

YTU, ÖNSÖZ (yildiz.edu.tr)

YBS Ansiklopedisi, https://ybsansiklopedi.com/wp-content/uploads/2015/08/Yazılım-Geliştirme-Modelleri-Yazılım-Yaşam-DöngüsüSDLCYBS.pdf

https://medium.com/databulls/bilgi-sistemi-uygulama-kontrolleri-e646627b22f9

Hızlı uygulama geliştirme (stringfixer.com)

https://acikbilim.yok.gov.tr/bitstream/handle/20.500.12812/89079/yokAcikBilim_10062556.pdf?sequence=-1&isAllowed=y