Yazılım Mühendisliği

Dr. Cahit Karakuş

Şubat-2018

İçindekiler

“Dinlediğini, okuduğunu, öğrendiğini anlamayan bir öğrenciden daha fazlasını istemek hiç de anlamlı değil.”

Tanımlar ve Kısaltmalar

Tanımlar:
İşlev (Fonksiyon): Girdi olarak kabul edilen değişkenlerden bir çıktı değer üretilmesini sağlayan kurallardır.

Süreç Yönetimi: İstatistiksel teknikler kullanılarak ve sağlıklı iletişim ortamı kurularak, farklı açılardan bakan kişilerin aynı dili konuşmasının sağlanmasıdır.

İstatistik: İş süreçlerinin gözlenmesi, bilgilerin sistematik olarak toplanması ve işlenmesi sonucunda belirli duyarlılıkta risklere yönelik tahminde bulunmayı ve yorum yapmayı sağlayan bilim dalıdır.

Süreç: İşlevlerin planlanıp değer ya da ürün haline getirildiği önceden belirlenmiş adımlardan oluşan iş akışıdır.

Değer: Üretilen ürün, teknoloji, hizmet, fikir, kültürdür. Günümüzde bilgi üretilmesi, bilinç üretilmesi, kültür üretilmesi, sanat ve felsefe üretilmesi, bilgelik üretilmesi katma değeri çok yüksek değerlerdir. Çünkü bunlar tarihi etkileyecek değerlerdir.

Paydaş: Süreçleri etkileyen ve süreçlerden etkilenen menfaat sahipleridir. Çalışanlar, yatırımcılar, kurucu sermayedarlar, hissedarlar, müşteriler, tedarikçiler, dış işlik alımları, iş ortakları, kamu kurum ve kuruluşları, bankalar ve finans kuruluşları, sendikalar, danışmanlar gibi tekneden ekmek yiyecek olanlardır.Paydaş, yazılımın başarısını etkileyen veya başarısından etkilenen kişi ve kurumlardır.

Kalite: Süreçlerin işlevlerinde toplanan verilere dayalı gerçekçi karar verme ve sürekli iyileştirme aşamalarında hataların temelinde yatan değişkenliğin belirlenmesi, azaltılması veya belli sınırlar dahilinde tutulmasıdır. Kalitenin temel kuramları olan ölçüm ve analize dayalı, sorgulama ve kıyaslama yapılarak hataların önlenmesi, maliyetlerin düşürülmesi, verimin artırılması hedeflenir.

Değişkenlik: Tüm olayların temelinde değişkenlikler vardır. Hataların, belirsizliklerin ve eksik verilerin çok büyük bir bölümü de değişkenlikten kaynaklanır. İstatistiksel teknikler kullanılarak değişkenliğin özellikleri incelenmeli ve hataların kaynakları tespit edilmelidir.

Toplam Kalite Yönetimi: İş üretilirken insan, sistem, yönetim ve ürün kalitesinin bir arada sürekli olarak geliştirilmesi, planlanması, kontrol çalışmalarının yapılması ve standartların oluşturulmasıdır.

Sürekli İyileştirme - Kaizen: Sürekli eğitim, sürekli bilgi ve beceri kazanma, sürekli araştırma ve geliştirme çalışmalarının yapılmasıdır. Organizasyonda iyileştirme çalışmalarının sürekli olması gerekir.

İstatistiksel Süreç Kontrolü: Organizasyonda sürekli gelişme sağlamak ve kaliteyi artırmak için iş üretilirken kontrol teknikleri için istatistiksel verilerden yararlanmaktır.

Benchmarking - Kıyaslama: Süreçlerde geliştirilen stratejilerin, örgüt yapısının ve tüm uygulamaların kıyaslanarak en iyisini bulmaya çalışmaktır. Mükemmel ve en iyisi olmak için en mükemmelini ve en iyisini aramaktır. Ararken değişilir, en iyi ve en mükemmele dönüşülür.

Dışarıdan Hizmet Alım (Dış İşlik): Organizasyonun her işi kendisi yapmak yerine asıl faaliyet alanı dışındaki işleri dış firmalara yaptırmasıdır.

Esnek Üretim: Ürünün istenildiği zamanda üretilmesi için gerekli planların hazırlanmasıdır. Stok üretim yerine, istenilen miktarda istenilen zamanda üretmektir.

Tam zamanında üretim – Just In Time.

Yalın Organizasyon Modeli: Organizasyon yapısının sadeleştirilmesi ve basitleştirilmesidir. Dikey organizasyon yerine yatay organizasyon yapısının oluşturulmasıdır.

Kademe Azaltma: Organizasyonda işe karar veren ile uygulayan arasındaki kademelerin mümkün olduğu ölçüde ortadan kaldırılmasıdır. Gereksiz ve katma değer yaratmayan kademelerin, bileşenlerin ve iş süreçlerinin kaldırılmasıdır.

Otomasyon: Rutin işlerin mümkün olduğu ölçüde robotlara ve bilgisayar kontrollü makinelere yaptırılmasıdır.

Sürekli Eğitim: İşe uygun eğitimli ve bilgili eleman alınmakla yetinilmemesi; sürekli eğitime, bilgi ve beceri kazandırmaya önem verilmesidir.

Otokontrol: Ekip içinde çalışanların birbirlerini kontrol etmeleridir.

Çalışanları Güçlendirme: Çalışanlara yetki ve sorumluluk devredilmesi ve bu sayede çalışanların karar alma sürecine katılmasıdır. Ekip çalışmasına önem verilmesidir. Çalışanların motivasyonu ve ödüllendirilmesidir. Çalışanların organizasyonda pay sahibi olmasıdır.

Toplam Verimli Bakım: Malzeme, enerji, ekipman ve personel ile ilgili kayıpların ortadan kaldırılması için bakım ve onarım çalışmalarının planlı ve düzenli bir şekilde yapılmasıdır.

Kurumsal Anlayış: Makama, yeteneğe resmiyet içeren etkili bir yönetim biçimidir.

Kurumsal Hafıza: Geçmişi hatırlamak amacıyla, üretilen değerler ile birlikte iş süreçleri ve tecrübelerin saklanması ve süreçten sürece bilinçlenerek aktarılmasıdır. Belgeler aracılığıyla bilgi ve fikir birikimi elde edinebilmektir. Kurumun geçmişi ile geleceği arasında ortak bir hafızanın gelişmesine katkıda bulunmak amaçlanır.

Kısaltmalar:
SDL: Security Development Lifecycle
TSP: Team Software Process
CMM: Capability Maturity Model (maturity: olgunluk, ergenlik)

Giriş

Karmaşık, büyük, entegrasyon içeren yazılım projeleri,ciddi bir süreç yönetimini zorunlu kılmaktadır. Aksi durumda başarıya ulaşması mümkün değildir. Bir yazılımın kalitesi, üretilen yazılımda hiç hata olmaması, gereksinimlerin tam olarak karşılanması, hata ayıklama ve onarma sırasında geçen zamanın uzunluğu, onarma hızı gibi kriterlerle ölçülmektedir. Yazılımın kalitesi, yazılım ekibinin kalitesini de belirler. "Ayinesi iştir kişinin, lafa bakılmaz."

Değer üretme adımları:

- İhiyaç, fikir
- Araştırma, analiz; teknoloji, Pazar
- Tasarım, Simülasyon
- Prototip üretme ve test
- Gerçek saha testi
- Ön ürün üretme
- Ürün Mükemmelleştirme
- Test
- Müşteri teslim
- Eğitim
- Servis, bakım
- Rekabete karşı direnmed destek sürekliliği

Yazılımcı sürekli yazılımı iyileştirmek ister. Teknoloji, bilimsel çalışmalar ve yenilikler sürekli uygulanmak ister, müşteriden yeni beklentiler ve istekler hiç bitmez.

Kalite, ilkeler topluluğudur. Öte yandan, yazılım geliştirme yaşam döngüsünde süreçler önemlidir ve sürekli iyileştirilme hedeflenmelidir. Standartlar belirlenmeli, sonuçlar sürekli değerlendirilmelidir.

Süreç, bir yazılımın gerçeklenme aşamasında değişkenliği azaltmaya ve iyileşme sağlamaya yönelik yöntemlerin geliştirilmesidir. Süreçler mutlaka yazılı hale getirilip belgelenmelidir ve tekrarlı işlemlerdir. Her sürecin işlevlerinin değerlendirildiği girdileri ve çıktıları vardır.

Yazılım süreci standartları:

- ISO 9001
- ISO/IEC 12207, ISO/IEC 15504 (SPICE)
- Carnegie Mellon Üniversitesi bünyesinde faaliyet gösteren Yazılım Mühendisliği Enstitüsü'nün (Software Engineering Institute - SEI) ortaya koyduğu CMMI (Capability Maturity Model Integration)

ISO / TL9000, süreçlere ve işlevlere uyum standardıdır. CMMI ise bir süreç iyileştirme modelidir. Yazılım mühendisliğinin sahip olması gereken özellikler:

- Yazılım Mühendisliği İlkeleri
- Yazılım Projeleri Yönetimi
- Yazılım Geliştirmede Yaşam Döngüsü
- Yazılım Geliştirmeyle İlgili En İyi Uygulamalar
- Güvenli Yazılım Sistemleri Geliştirme
- Yazılım Testi
- Yazılım Mühendisliği Yöntem ve Teorisi
- Yazılım Mühendisliği Etik Kuralları
- ISO / IEC Yazılım Mühendisliği Standartları
- Yazılım Mühendisliğinin Geleceği
- Oyun Tasarımı Yazılım Mühendisliği

Yazılımlardaki zafiyetlerin yaygınlığı ve çokluğu, zaman ve maliyet kriterlerini sağlamakla beraber, büyük ölçüde güvenliğe odaklanan daha sıkı bir yazılım geliştirme sürecine gereksinim olduğunu göstermektedir.

Geliştirilen bir yazılım projesinin planlanmasından başlayarak teslimatına kadar geçirmiş olduğu bütün aşamalara ve bu aşamalardan oluşan döngüye yazılım geliştirme yaşam döngüsü denir. Basitçe bir proje geliştirilirken projenin planlama, analiz, tasarım, gerçekleştirim ve bakım aşamaları yer almaktadır. Yazılım projelerindeki başarısızlık oranlarına bakıldığında istenen sonucun elde edilemediği görülmektedir. Yönetim sürecinden kaynaklanan sıkıntılardan dolayı yazılım projelerinin başarı oranı, %10 - %30 arası başarıyla çok düşük seviyelerdedir.

Yazılım projelerini yönetmek için çevik yöntem olarak bilinen yeni yöntemlerin gerekliliği ortaya çıkmıştır. Çevik yöntemler tekrarlı olarak ilerler, gereksinimlerin ve tasarımın kademeli olarak ortaya çıkmasına dayanır, şelale süreç modelinin yoğun yazılı dokümantasyonundan çok, doğrudan yüz yüze iletişime vurgu yapar. “The Standish Group” tarafından 2015 yılında yayınlanan “Chaos” raporunda, 2011-2015 yılları arasında çevik ve şelale yöntemlerin uygulandığı projelerin başarı oranlarının karşılaştırıldığı çalışma, çevik yöntemlerin daha başarılı olduğunu göstermektedir.

1. Yazılım Mühendisliği

1.1. Yazılım

Yazılım mühendisliği ilkelerine uyularak daha iyi tasarlanmış yazılım.

Yazılım, herhangi bir boyuttaki herhangi bir tür donanımda çalışan bilgisayar programını, basılı veya elektronik ortamdaki her tür dokümanı içeren ürün.Dokümanlar yazılım geliştirme ve son kullanıcıya yönelik olabilir.

Yazılım bir üründür, ancak başka ürünler geliştirmeye veya elde etmeye yarayan bir araç da olabilir.Yazılım fiziksel bir ürün olmadığı için aşınmaz, ancak zamanla yetersizleşebilir.

Yazılım bir üründür, ancak başka ürünler geliştirmeye veya elde etmeye yarayan bir araç da olabilir.
Yaşam döngüsü: Yazılımın bir fikir olarak doğmasından, kullanım dışı bırakılmasına kadar geçen süreç.

Yazılım fiziksel bir ürün olmadığı için aşınmaz, ancak zamanla yetersizleşebilir.Değişim kaçınılmazdır: Yazılım, yaşam döngüsü süresince değişikliklere uğrar.Değişiklikler, yazılımda yeni hatalar oluşturabilir.Yeni hatalar tam olarak düzeltilmeden yeni değişiklikler gerekebilir.

Yazılımı etkileyen eğilimler

- Yaygınlaşan Bilgi-İşlem: Hesaplama gücünün giderek küçülen alanlara sıkıştırılabilmesi, bilişimin günlük yaşantımızla daha kolay bütünleşmesine olanak sağlıyor.
- Yaygınlaşan Haberleşme Ağı: Kablosuz ağların yaygınlaşması, bilişimin günlük yaşantımızla daha kolay bütünleşmesine olanak sağlıyor.
- Özgür / Açık Kaynak Yazılım: Gevşek bir ekip tarafından geliştirilen yazılım, daha anlaşılır ve geliştirilebilir olmalıdır.
- Takım çalışması zorunluluğu
- Küreselleşme,
- Ekonomik krizler

1.2. Yazılım Türleri

- Sistem Yazılımı: Diğer programlara hizmet sunmak üzere hazırlanmış programlar, derleyiciler, işletim sistemleri.Derleyiciler, işletim sistemleri, vb.
- Mühendislik Yazılımı / Bilimsel Yazılım: Mühendislik ve bilimsel hesaplamalarda kullanılmak üzere hazırlanmış programlar.Büyük hacimli verilerle uğraşır. Rakamları öğütürler (Number crunching).
- Gömülü (Embedded) Yazılım : Donanım ile çok sıkı ilişkidedir.Denetim amaçlıdır.Gerçek zamanlı uygulamalardır.
- Uygulama Yazılımı: Product-line, shrink-wrapped, (commercial) off-the-shelf,vb.Bir çok mühendislik alanında olduğu gibi Yazılım Mühendisliği alanında da tanımlanmış TS/BS ISO/IEC 25051 COTS standartları vardır. Farklı müşteriler tarafından kullanılabilecek genel amaçlı yazılımlar. Cari hesap uygulamaları, çeşitli otomasyon programları, kelime işlem uygulamaları, vb.
- Kurumsal Yazılım: Belirli ticari iş gereksinimlerine yönelik programlar. İş süreçleri ile ilgili bilgiye sahip olmalıdır.Genellikle müşteriye özel tasarlanır.Veri dönüştürme ve değerlendirme uygulamaları, iş süreçlerinin kimi zaman gerçek zamanlı izlenilmesi, vb.Zamanla "eski yazılım" haline dönüşür!

ESKİ YAZILIM (Legacy Software):

İş sürecinin önemli bir parçası olan ve çok uzun süredir kullanılan yazılımlar.

- Eski yazılımda bulunabilecek olumsuzluklar:Eksik veya hatalı dokümantasyon, Zamanla karmaşıklaşmış kod, Esnek olmayan yapı, Eski donanımla çok sıkı ilişki,Yazılım mühendisliğindeki gelişmelerden yoksunluk nedeniyle düşük kalite.
- Eski yazılımın değiştirilmesini gerektiren nedenler:İş alanındaki yeni gereksinimler. Güncel sistemlerle birlikte çalışabilmesi için uyumluluk kazandırılması. Donanımın ömrünün dolması nedeniyle daha güncel ortama taşınma gerekliliği.

1.3. Yazılım Mühendisliği Hakkında Yanılgılar

Programcı açısından yazılım hakkındaki yanılgılar
Yazılımı tamamlayıp müşteriye teslim edince işimiz biter. Oysa yazılım üstünde harcanan çabanın yarısından fazlası, yazılımın müşteriye ilk teslimatından sonra harcanmaktadır.

Yazılımı tamamlamadan kalitesini ölçemem. Oysa kalite güvence yöntemleri yazılım hayat döngüsünün her aşamasında uygulanabilir.Çözümleme sürecinde dahi kullanılabilecek kalite ölçütleri bulunmaktadır.

Yazılım eşittir program. Oysa gereksinim analizi başlı başına bir emektir.Dokümantasyon ve sınama çalışmaları da unutulmamalıdır. Bazı durumlarda entegrasyon çalışmaları da gerekmektedir.

Yazılım mühendisliğinin gereklerini uygulayarak boşuna çaba harcıyoruz. Oysa haritası olmayan yolunu kaybeder.Kalite için harcanan çaba, karşılığını yazılım hayat döngüsünün ilerleyen aşamalarında fazlasıyla ödeyecektir. Küresel ölçekte yazılım projelerinin yarısından fazlası başarısızlığa uğramaktadır.

Müşteri açısından yazılım hakkındaki yanılgılar
Programın yazılmasına başlanması için amaçları genel olarak belirlemek yeter, ayrıntılar sonra kararlaştırılabilir. Oysa belirsiz gereksinimler, çürük atılmış temele benzer.

Yazılım esnektir. Değişen gereksinimler kolayca sisteme uyarlanabilir.Oysa yazılım yaşam döngüsünde ilerledikçe, değişen gereksinimleri yazılıma uyarlamanın bedeli üstel olarak artar.
Yazılım esnek bir oyun hamurundan çok kil veya cam gibidir.Çevik süreçlerle esnekliğin arttırılması hedeflenmektedir.

İdari açıdan yazılım hakkındaki yanılgılar
İşler yetişmiyorsa takıma yeni programcılar ekleriz. Oysa yazılım hayat döngüsü içerisinde ilerledikçe, yeni elemanların yazılıma hakim olması üstel olarak zorlaşır. İşler daha da gecikir.

Geliştirmesini üstlendiğim yazılımı tamamen veya kısmen fason yaptırırım.Proje ilerlemesini kendi içinde denetleyemeyen bir firma, dışarıya verdiği işi izlemekte de zorlanacaktır.

Açık kaynak yazılım üretirsem kar edemem.Danışmanlık hizmetleri ile kar edilebilir.Başka iş modelleri de vardır.

2. Değişimi Yönetmek

Sosyalleşmede, teknolojide, üretimde, çevrede, uzayda, ... değişim baş döndürcü bir hızla devam etmektedir. Yazılım mühendisliği olarak bu değişimin ya içerisinde olacağız, ya izleyeceğiz, ya da farkında bile olmayacağız. Aslında değişim adım adım sizi kapsar. Değişimin karşısında kendisini gösteren direnme, ret etme gibi ortamdaki bariyerler anlamsızı anlamlı kılar, ve ipek böceği kozası gibi hapseder. Örümcek ağına düşersiniz, farkında olmanız imkansızdır, donuklaşırsınız. Dışarısı çok hızlı değişmektedir; fark edilmez durumu, yok oluşu başlatır. Değişim inanmaktır, kaçınılmazdır. Değişimin ne olacağı öngörülür fakat nasıl olacağı ve ne zaman olacağı öngörülmez. Değişimin etkileri analiz edilebilir.

Değişim: Bir süreçtir, aniden olmaz. Değişim akılla yapılmaz. Değişime direnç olmazsa olmazdır. Değişim kişilerin süreci nasıl algıladıklarına bağlıdır. Başarı öyküleri en büyük motivasyon aracıdır. Değişim için geliştirilecek model duruma bağlıdır.

Değişimin hızı ve maliyeti iyi analiz edilmelidir. Herkes değişimden bahseder. Değişim yönetilirken öncelikle bariyerler sonlandırılır. Değişim fark edildiğinde, hata yapmaktan korkulmamalı, ders alınması ve tekrar edilmemesi önemsenmelidir. Çünkü hata yapmak en?iyi öğrenme metotlarından birsidir. Değişim sırasında lider, motive eder, hayal kurdurur, destek verir, cesaretlendirir, sürece yön verir, değişimin hızını korur.

Değişim türleri:

- Teknik değişim: Modeller ve roller, elde edilecek gelir, görsel özellikler.
- Kişisel değişim: Müşteri, yatırımcı, beklenti, ekip, rakipler, gelirler, bakış açısı.
- Bilgi ekonomisi değişimi tetiklemektedir; sanal, e-ticaret, dijital.
- Ekip: Birlikte çalışma, değer üretme, ticarileştirme, vaz geçme.

Değişimde alışkanlıklardan nasıl vazgeçeceğiz? Harika ile başlayan cümlelerde duyguyu satın alırsınız.Direnci kırmada sonuç ve çıktıların kendileri için iyi olacağına inanılması etkin rol oynar. Değişim başladığında ekip, kültür ve lider olarak önce mevcut davranış ve alışkanlıklar sonlandırılır, hayal kurulur. “Ne yapabiliriz?”, “Nasıl yapabiliriz?” keşfedilir. Yeniden başlanır. Değişimin gerekçeleri tespit edilir. Nerede olduğunuz ve 3 yıl sonra nerede olacağınız hayal edilir. Soru sorulur: Neyi değiştirmek istersiniz?

Sorun, teknoloji üreten insanlarla ürettikleri nesneler arasındaki ilişkidedir. Dünyayı düzeltmenin yeri önce kendi yüreğimiz, kafamız, ellerimiz ve onlardan çıkan iştir. Çünkü insanoğlu ürettiklerini düzeltmek yerine nasıl onarılacağına odaklanmalıdır.

Etkin değişimin beşlisi:

- Motive etmek
- Değişimi hayal ettirmek
- Grup içi destek oluşturmak
- Değişimin adımlarını yönetmek
- Hızı korumak

Değişimde motivasyon için:

- İç dinamikteki değişim gereksinimleri gözlemlenir.
- Olan ile olmak istenen yer arasındaki fark tespit edilir.
- Değişim sonrası olabilecek başarı öngörülür.
- Değişim gerektiren dış ve iç etkenler tespit edilir.
- Dış ve iç etkenler abartılır.
- Sıkıntı yaratan etkenler azaltılır ya da ortadan kaldırılır.
- Olası değişim inancı için ortak yön bulunur.
- Değişimin başladığı gözlemlenir.

Değişimin bariyerleri:

- Değişime direnç
- Sistemdeki kısıtlamalar
- Yöneticilerden destek görmeme
- Sponsor eksikliği
- Gerçeklikten uzak beklentiler
- Farklı profillerin olmaması
- Yetersiz ekip ya da yetenek
- Teknik ekiplerin katılmaması
- Konunun çok dar olması

Organizasyonel direnç güç odaklıdır, eski ilişkilere dayanır, yapısaldır, değişime yeterince odaklanma olmamıştır, ekipseldir. Direnç! inkar ile başlar. Söylemler başatılır; "eskiden ne güzeldi", "burada bu adamlarla olmaz o iş", uyuşukluk, "hiçbir şey değişmedi". Yeni bilgiyi dinlememenin sonucunda direnme fiziksel bir hal alır. Direnme başladığında öfke, şikayet ve suçlama inanılmaz boyuttadır.Direnen kişide yaralı, kaybetmiş ve inatçı kişilik belirtileri başlar. Karşısındakini test eder. Herşeyi çok iyi bildiğine, karşısındakinin hiçbirşey bilmediğine inanır.

Direnci aşmanın yolları: eğitim ve iletişim, kolaylaştırma ve destek olma, tartışma ortamında beklentileri karşılama, güdümleme ve zorlama olarak sıralanır.

Değişimin hızını korumak için:

- Yeterli kaynak ayırılmalı
- Değişimden sorumlu kişiler! belirlenmeli
- Yeni yeteneklerin gelişmesine olanak sağlanmalı
- Kişi, yeni davranışları etkin kullanıma zorlanmalı
- Rotadan sapma! engellenmeli
- Değişim! motive edilmeli
- Gelecek vizyonu! çizilmeli, hayal edilmeli ve ettirilmeli.
- Değişim için grup oluşturulmalı.

2.1. Statüko ve Bariyerler

İş süreçlerinde değişim, gözardı edilmemesi gerekilen bir gerçektir. Mevcut durumun devam etmesi adına değişimlerin fark edilmemesi her türlü yıkıcı oyunları temsil eder.Statü mevcut durumun devamı demektir. Statükonun amacı değişime kapalı bir sistem yaratıp, bu sistemi vazgeçilmez bir sistem haline getirmektir. Çünkü statüko çıkar üzerine kurgulanmıştır. Statüko, değişime! yok olma pahasına direnmektir.

Karar veren ile işi yapanın aynı paydada olmaması durumunda işi yapan işini yüklenmez ve inanmaz ise proje yürümez. Direnç inkar ile başlar, ihtiyacım yok ile yoluna devam eder.

- Değişim olurken direnme süreklilik kazanma çabası iç!erisindedir; hiçbirşey değişmediği sürekli tekrarlanır, öneri ve bilgi dinlemez.
- Sıkıştığında tamam der, sonrasında sürekli bahane üretir; sık sık hastalanır, sizi test eder, yaralı ve kaybetmiş rolünü oynar.
- Bahaneleri sıralar; para yok, ekip yok, zaman yok...
- Sonra yavaş yavaş "Olabilir mi?" sorusunu sorgulamaya başlar. (Kendi kaosunu oluşturur.) Bu aşamada geleceğe tutundurarak işe başlatmak gerekir.

Değişim yaratılacaksa dirençler kırılmalıdır. Direnç ! inkar odaklıdır. Dirençler kırıldığında ortam enerjik olur ve alternatifler aranmaya başlanır. Bu durum odaklanamama ve kaos olarak kendini gösterir. Sonrasında işbirliği ve dengeler oturmaya başlar ve ekip oluşur.

Örgütsel değişim ancak çalışanların katılımı ve desteklemesiyle başarılabilir. Eğer örgütsel direnç kısa bir süre! içinde ortadan kaldırılmazsa; örgüt, enerjisinin büyük bir bölümünü örgütsel değişim yerine! direnmenin doğurduğu sorunlara harcamak zorunda kalır. Böylece örgütsel değişim yönetimi başarısız olur.

Değişim, çalışanların tüm dengelerini altüst edebilir; bu nedenle çalışanlar değişime direnç başlatır. Psikolojik direnmede değişime olan kızgınlık önemsenmelidir. Kızgınlığı tetikleyen faktörler: belirsizlik ve bilgi eksikliği, yabancılaşma, alışkanlıklar, güvensizlik, ilgisizlik, peşin hüküm, algılama biçimi, kişisel düşmanlık, değişimin yanlış olduğu inancı, farklı değerlendirmeler ve hedefler, empoze edilen değişimden memnuniyetsizlik, değişimin getirdiği yeni kurallar ve kontrol artışına karşı duyulan kızgınlık şeklinde sıralanabilir. Teknolojik değişikliklerde direnç oldukça yaygın biçimde görülür.

İletişim kanallarının açık olması, yönetim ve astlar arasında karşılıklı güvenin oluşmasına ve aynı zamanda değişime karşı duyulan endişenin ve direncin azalmasına yardım eder. Bilgi, beceri ve yetenekleri eğitim yoluyla geliştirilen birey ya da grupların değişime karşı dirençleri azalır. Bir başka deyişle değişimin gerektirdiği yeni bilgi ve beceriye sahip olan çalışanlar değişime direnmek yerine onu arzu eder bir hale gelirler.

Bazen değişime direnme kuruma fayda da sağlayabilir. Yanlış geliştirilen uygulamanın süreçleri olumsuz etkileyeceğini fark eden çalışanlar haklı direnç gösterirler.

2.2. Güçlü ve Zayıf Yönlerin Belirlenmesi

Güçlü ve zayıf yönler doğru belirlenirse, oluşabilecek fırsatlar ve tehditler de hızlı fark edilir. Güçlü yönler amaca ulaşmada başarıyı, zayıf yönler ise aşılması gereken engelleri gösterir. Belirlenecek güçlü yönler hedeflere, zayıf yönler ise tedbirlere ışık tutar. Önce güçlü ve zayıf yönler ortaya konmalı, sonra da rakipler karşısındaki durum kıyaslanarak sapmalar belirlenmelidir.

Güçlü ve zayıf yönler belirlenirken;

a) Personel değişikliklerinin neden olacağı **yetki çatışmaları** belirlenir.
b) Paydaşların, organizasyon kültürüne katkı sağlayacak yetkinlikleri ve deneyimleri saptanır.
c) Organizasyonun kültür seviyesi belirlenir.
d) Teknolojik alt yapı ve çalışanların teknolojiyi kullanma düzeyinden teknoloji indeksi elde edilir.
e) Mali kaynaklar, bütçe, malzeme dökümü ve diğer varlıklardan mali durum tanımlanır.
f) Ekonomik, politik, çevresel ve teknolojik gelişmelerin rekabete yönelik etkileri saptanır.
g) Dünya ve ülkedeki değişimler, gelişmeler, eğilimler ve kritik konuların kurumu nasıl ve ne yönde etkileyeceği sürekli izlenir.
h) Kurumun karşılaşabileceği riskler ve belirsizlikler sistematik olarak takip edilir ve raporlar hazırlanır.

i) İş süreçlerinde uygulama aşamaları tamamlandığında, sonuçların amaç ve hedeflere tutarlılığı ve uygunluğu değerlendirilir.

Pazardaki fırsatlar ve tehditler doğru tespit edilmelidir. Organizasyonda iç ve dış durumların değerlendirilmesinde "SWOT Analizi" kullanılır. **SWOT** İngilizce **S**trength (güçlü yönler), **W**eakness (zayıf yönler), **O**pportunity (fırsatlar), **T**hreat (tehtid ve tehlikeler) kelimelerinin baş harflerinin birleştirilmesinden oluşur.

İzleme ve değerlendirme faaliyetleri sonucunda elde edilen bilgiler kullanılarak, var olan plan gözden geçirilir. Hedeflenen ve elde edilen sonuçlar karşılaştırılarak hesap verme sorumluluğu oluşturulmalıdır.

Başarı performansının ölçülmesi ve değerlendirilmesinde amaca uygun doğru ve tutarlı bilgiler elde etmek için;

a) İhtiyaçların temin edileceği kaynaklar ve nakliyesi, depolama yerleri, teslim süreleri ve maliyetleri
b) Alternatif teknolojiler, üretim süreçleri, teslim tarihleri ve gelecekteki olası gelişimleri
c) Yedek parçalar, bakım onarım araç ve gereçleri
d) Yer seçiminde çevresel etkiler ve çevre koruma önlemleri
e) İnşaat işleri, çevre düzenlemeleri
f) İş kazası oluşturacak risklere karşı alınacak güvenlik tedbirleri
g) İnsan gücü, nitelik düzeyleri ve maliyetleri ile birlikte ilave yükleri
h) Tesis, mali, üretim, satış ve idari organizasyonları
i) İş süreçlerinin önerilen zamanlamada ve maliyetlerde gerçekleştirilmeleri
j) Sermaye yapısı ile finansman tabloları, kaynakları, planlaması ve oranları
k) İş akışında ilk yatırım, işletme-bakım-onarım –yenileme harcamaları ve analiz edilmeleri

dikkate alınacak konular arasındadır.

İş süreçleri uygulama aşamalarını olumsuz etkileyecek harcamalar belirlenmeli, yatırımın başlangıcından sonuna kadar ekonomik analizler zaman periyodunda sıralanıp (fayda-masraf) farkları hesaplanarak para akış tablosu oluşturulmalıdır. Para akış analizinin doğru yorumlanabilmesi için bütçe analiz periyodu boyunca harcanan para akışının, belirlenen oranlara göre yatırımın ilk başlangıç tarihine indirgenerek toplamlarının alınması suretiyle net bugünkü değer hesaplanmalıdır. Projenin ticari analiz hazırlama etütlerinde, net bugünkü değer, iç karlılık oranı, fayda-maliyet oranı, geri ödeme süresi, basit karlılık oranı, başa baş noktası, duyarlılık analizi, katma değer etkisi, net döviz kazancı, istihdam etkisi gibi temel hesaplamalar yapılmalı ve işletme dönemi gider - gelir raporları hazırlanmalıdır. Toplam yatırım tutarı hesaplanmasında etütler yapılırken sabit sermaye yatırımı, işletme sermayesi ve yatırımın zamana dağılımı hesaplanmalıdır. Sermaye girdilerinin elde edilebilmesi için gerekli sabit ve aylık giderler hesaplanmalıdır.

2.3. Murphy Kanunu

Murphy Kanunları , Amerikalı mühendis Edward A. Murphy, Jr. tarafından, başarısızlıklar ve hata kaynaklarının karmaşık sistemlerde incelenmesi üzerine ortaya konan özdeyişlerdir. Kuralların ortaya çıkışı 1949'lara dayanıyor. O dönem askeriyede mühendis bir yüzbaşı olarak görev yapan Murphy, bir çarpışma testinde her seferinde hata çıkmasına sinirleniyor ve hata yapan elemana ithafen "Eğer bir işi yanlış yapmanın bir yolu varsa, bu adam onu mutlaka bulur." diyor. Ve bu lafından sonra takım arkadaşı olan John Stapp tarafından meşhur edilmesi süreci başlıyor. John Stapp, Murphy'nin bu laflarını kendi arkadaş ortamlarında eYazılım Problem Raporlamaili bir dille dile getiriyor ve kendi aralarında "Murphy Kanunları" diye bir oluşum yaratmaya başlıyorlar. Daha sonra John Strapp, çarpışma testi projesi için röportaj verdiğinde; "Yıllardır yaptığımız testlerin güvenilirliğini Murphy kanunlarının bize gösterdiği sonuçlara borçluyuz." demiştir. Ve böylelikle Murphy kanunları dünya genelinde yaygınlaşmıştır.

Murphy Kanunu: "Eğer bir işin birden fazla sonuca ulaşma olasılığı var ise ve bu olasılıklardan biri istenmeyen sonuç veya felaket doğuracaksa; birşeyler ters gitmeye başladığında kesinlikle en kötü olasılıklar sıralı gerçekleşecektir." Analitik ölçüt olarak hataları önleme stratejisi olarak kullanılır.

Olasılık, gerçek sonuçların olası sonuçlara oranı şeklinde tanımlanır. Bir olayın gerçeklenme olasılığı düşükse gerçekleşme olasılığı da düşüktür, ya da imkansızdır. Araba ile Ankara'ya giderken kaza olma olasılığı çok düşüktür yani kaza geçirme olasılığının imkansız olduğunu düşünürsünüz. Murphy Kanunları ise olaya tersinden yaklaşır: Bir olay mümkünse, gerçekleşir. Basit bir cümleyle söylemek gerekirse; kaos, düzenden daha olasıdır.

Murphy kanunları nelerdir?
Üzerine reçel sürdüğünüz ekmek yere düştüğünde, her zaman reçelli yüzü yere dönük olacaktır.
Çözülen her bir problem, yeni problemler yaratır.
Bozuk bir alet tamire geldiğinde mutlaka çalışır.
Anlamıyorsanız çok açıktır.
Her şey mükemmel gidiyorsa, mutlaka bir yerde bir terslik vardır.
Düşman menziline girdi diye sevinme, sen de onun menzilindesin.
Savaşta ilk önce ölenler hiç korkmayanlardır. Onları en çok korkanlar takip eder.
Fark edilmediğinizi düşündüğünüz zamanlarda, herkes tarafından izleniyorsunuzdur.
Pimi çektiğiniz an, Bay El Bombası artık arkadaşınız değildir.
Bir bölgeyi güvenlik altına aldıysan, bunu düşmanına söylemeyi unutma.

Anonim anlatı:
Duvardaki çatlaktan bakan fare, çiftlik sahibi ile karısının bir paket açtıklarını gördü. "İçinde yiyecek mi var?" derken, bir baktı ki fare kapanı!! Hemen bahçeye koşup alarmı verdi: "Evde kapan var! Evde kapan var!" Tavuk gıdaklayıp, kafayı kaldırdı: "fare, bu sizin için ciddi bir sorun olsa da, beni ilgilendiren bir tarafı yok ne yazık ki!" Koyun konuyla ilgilendi ama kendi hesabına: "Üzgünüm fare,

vah vah emin ol senin için dua edeceğim" dedi. Öküz: "Fare, Senin için üzüldüm, ama burnumu sokacağım bir şey değil." dedi. Fare yalnızlık ve terkedilmişlik hisleri içinde, kendisini enseleyecek fare kapanı ile artık tek başına başa çıkmaya çalışacaktı!

O akşam evde, alışılmamış bir ses duyuldu. Sanki bir kapan, avının üzerine kapanmıştı. Sese koşan çiftçinin karısı, karanlıkta kapana zehirli bir yılanın kuyruğunu kaptırdığını görmemiş. Yılan da kadını ısırmıştı. Çiftçi karısını hemen hastaneye götürdü. Karısı eve ateşli ve hasta olarak döndü.

Yüksek ateşli insana ne içirilir? Sıcacık bir tavuk çorbası!!! Tavuk hemen kesildi, pişirildi!
Ama kadın hala iyileşmiyordu. Eş dost ahbap hasta ziyaretine gelince, çiftçi de sofraya koyunu çıkarmak zorunda kaldı! Derken çiftçinin karısı iyileşmedi ve öldü! Aman ne kalabalık gelmiş cenazeye, ne kalabalık! Bu sefer de konukları doyurmak için kesilen öküz oldu. Fareye de olan biteni deliğinin ardından izlemek kaldı!

Bir işlev yerine getirilirken kötü bir şeyin olma olasılığı çok düşük olabilir; önemli olan istenmeyen olay gerçeklendiği anda oluşacak tüm kötülüklerin olma olasılığının çok yüksek olmasıdır.

2.4. Akıl Oyunları

Karar verme, karar vericilerin problemleri tanımlaması ve çözüm alternatifleri arasından seçim yapması sürecidir. Oyun kuramı, insanların karar verirken etkileşim içinde oldukları ile işbirliğinin modellendiği bir yaklaşımdır. Oyun, gerçek bir işletme probleminin ya da durumunun özet biçiminde modellenmesiyle ortaya çıkmaktadır. En çekici görünen seçeneklerden uzak durduğunuz zaman kaybettiğiniz şeylerin karşılığında daha az zarara uğrayabilirsiniz. Doğal olarak hedefiniz sadece beklenmedik biçimde davranmak değil, bunu belli bir olasılık stratejisine uygun olarak yapabilmektir.

Günümüz rekabet koşullarında daha etkin kararların verilmesinde rasyonel karar verme sürecinin izlenmesi ve çeşitli karar verme modellerinin kullanılması bir gereklilik olmaktan çok bir zorunluluk haline gelmiştir. Oyunu bilmek kazanmayı garantilemez, strateji seçim, etkileme konusunda düşünme yapısı kazandırır. Bu nedenle oyun, oyuna katılanların çatışma altında amaçlarını elde etmeye çalışırken birbirlerini engellemeye çalıştıkları bir ortamda oynanabilir. Oyun teorisindeki ünlü 'Nash dengesine' adını veren dahi matematikçinin yaşamının 'denge' üzerine kurulduğunu söylemek zordur.

Oyun kuramında Nash dengesi,

- Karar verenlerin diğer düşüncelerle uyumlu ya da rekabet halinde olduğu durumları modeller.
- Oyun kuramının geleneksel uygulamaları bu oyunlarda —bireylerin davranışlarını değiştirmek istemediği— denge bulmaya çalışır. Davranışın değişmek istemeyeceği denge bulmaya çalışılır.
- Oyun kuramı fikrinde gerçekleştirilmek üzere bir çok denge kavramları geliştirilmiştir. En ünlüsü Nash dengesidir.
- Denge kavramları uygulama alanına göre farklı amaçlara sahiptir, fakat genel olarak uyuşurlar ve iç içe geçmişlerdir.
- Akıl oyununa ilişkin yöntemler eleştiriden uzak değildir ve bazı özel denge kavramlarının uygunluğu, dengenin tümden uygunluğu ve genel olarak matematiksel modellerin faydaları üzerine tartışmalar sürmektedir.
- Oyunların çoğu karşılıklı etkileşim ve rekabet içerisindedir. Oyuncu oyundaki diğer oyunculardan üstün olmak için çabalar ve başarısı diğerlerinin ve kendinin seçimlerine bağlıdır. Birinin kazancının ötekinin zararına olacağı yarışmaları çözümlemek için geliştirilen oyun teorileri birçok kıstasa dayanır ve etkileşim alanları detaylı incelenmelidir.
- Akıl oyunları kuramının en temel özelliği, karar verenlerin düşüncelerini ve rekabet halindeki sosyal durumlarını modelleyen bir yaklaşım olmasıdır.

Karar verenlerin diğer düşüncelerle uyumlu ya da rekabet halinde olduğu durumları modelleyen bir yaklaşım olması bu kuramın en temel özelliğidir.

Oyun kuramının geleneksel uygulamalarında denge bulmaya çalışılır. Bu fikri gerçekleştirmek üzere birçok denge kavramları vardır. En ünlüsü Nash dengesidir. Oyun kuramları eleştiriden uzak değildir ve bazı özel denge kavramlarının uygunluğu, dengenin tümden uygunluğu ve genel olarak matematiksel modellerin faydaları üzerine tartışmalar sürmektedir.

Oyun kuramı ve kişilerin riske karşı tutumları incelenirken fayda beklentilerinin karar üzerindeki etkisi göz ardı edilmemelidir. Etkileşimli karar vermede, ekip içerisindeki her bir kişinin ödemesi gereken bedel ve elde edilecek fayda kararlarına yansır. Karar vericiler bir ekibin oyuncularıdır. Seçimleri diğer kişilerin tercihlerine bağlı olarak farklılık gösterir. Bu durumda taraflar davranış biçimleri ile ilgili olarak birtakım kurallar üzerine anlaşmayı tercih edebilirler. Dolayısıyla her karar bir risk içerdiğinden etkileşimli karar ortamında ya da çatışma altında karar verme durumunda ekip üyelerinin riske karşı tutumları önem kazanmaktadır.

Oyunların çoğu karşılıklı etkileşim ve rekabet içerisindedir. Oyuncu oyundaki diğer oyunculardan üstün olmak için çabalar ve başarısı diğerlerinin ve kendinin seçimlerine

bağlıdır. Birinin kazancının ötekinin zararına olacağı yarışmaları çözümlemek için geliştirilen oyun teorileri birçok kıstasa dayanır ve etkileşim alanları detaylı incelenmelidir.

Akıl oyunları kuramının en temel özelliği, karar verenlerin düşüncelerini ve rekabet halindekilerin sosyal durumlarını modelleyen bir yaklaşım olmasıdır. Amaç size önerilen fayda modelinin size vereceği zararı sorgulayarak bulmaktır. Bir fareye; "Eğer bulunduğun delikten çıkıp yandaki deliğe girersen sana büyük bir kalıp kaşar peyniri verilecektir." denir. Fare sorgulamaya başlar; "Mesafe çok küçük, rüşvet aşırı ve abartılı büyük, bu işte bir puştluk olmalıdır." der. Çıkar paylaşımlarına dayanan günümüz ilişkilerinde paylaşma oranları, ilk başta kabul edilse bile süreç içerisinde taraflar sorgulamaya başlar. Bu da paylaşımcılar arasında çatışma meydana getirir.

Birden fazla sığınağın bulunduğu bir savaş alanında bir askerin tepesinde daireler çizen uçağın içerisinde bomba bırakmak için fırsat kollayan bir pilot düşünün. Normalde askerin çevredeki en sağlam görünüşlü sığınağı seçmesi ve orada saklanması gerekir. Aynı anda pilot da askerin en iyi sığınağı seçeceğini düşünerek orayı bombalamak isteyecektir. Bunu bilen askerin o denli sağlam görünmeyen ikinci sığınağı seçmesi gerekmez mi? Eğer ikisi de çok akıllıysa, olasılıklara dayanan stratejiler izlerler. Örneğin asker çevredeki çeşitli sığınaklar arasından ona en fazla kurtulma şansı verecek özelliklere sahip olanları arar, bundan sonra nereye saklanacağını belirlemeye çabalar. Pilot da askeri vurma şansının en yüksek düzeyde olduğu sığınağı belirlemek için benzer biçimde olasılıklardan yararlanır. Bu saçma gelebilir ama ikisi de akılcı davranabiliyorsa yapacakları budur. Doğal olarak asker hareketlerini gizlemezse pilotun işi kolaylaşır, buna karşılık pilot da nereyi bombalamayı tasarladığını askere sezdirmemeye çalışmalıdır.

Nash dengesinde amaç öngörüde bulunmaktır. Kısaca öngörmektir. Nash, herhangi bir stratejik etkileşimde, bir oyuncunun en iyi seçiminin ya da hamlesinin, öteki oyuncuların ne yapacaklarına dair inancına sıkı sıkıya bağlı olduğunu fark etti. Nash, her oyuncunun, öteki oyuncuların yapabileceği hamle seçeneklerine bakarak en uygun hamleyi seçtiği duruma bakmamızı önerdi.

İki oyuncu, birbirinden bağımsız olarak ve aynı anda, 180'le 300 lira arasında bir miktar seçsinler. Bu oyunda seçilen en düşük miktar her iki oyuncaya ödenecektir. İki miktar arasındaki fark büyük miktarı seçenden alınacak küçük miktarı seçene verilecektir. İki oyuncu aynı miktarı seçerse, ikisine de seçtikleri miktar ödenecek, ayrıca transfer yapılmayacaktır. Örneğin seçilen büyük miktar 230, küçük miktar 200 olsun. Bu durumda küçük miktarı seçene 230, büyük miktarı seçene 170 ödenecektir. Sonuç olarak her ikisi de 180 i seçecektir. Eğer bu oyunda fark değil de belirlenen herhangi bir miktar, büyük miktarı seçenden alınıp küçük miktarı seçene verilecek olsaydı. Belirlenen miktarın büyüklüğü, karar vermede önemli rol oynayacaktır. Belirlenen miktar küçüldükçe seçilecek miktar 300'e doğru artacaktır.

Diğer bir örnekte kişilerden 0'la 100 arası bir sayı seçmeleri istensin. Seçilen sayıların ortalamasının yarısına en yakın olan oyuncu ödül kazanacaktır. Oyuncular hangi sayıyı seçeceklerdir? Bu oyunda bir oyuncu 0 ile 100 arasındaki her sayının seçilme olasılığının aynı olduğunu düşünsün ve 50'yi seçsin. Bir önceki oyuncun 50 seçeceğini düşünen diğer bir oyuncu 25 seçmesi gerektiği sonucuna varır. Bir sonraki oyuncu, ilk iki oyuncunun 50 ve 25 sayılarını seçeceklerini düşünüp 12 ya da 13 seçmeye yönelecektir. Ekonomistler tarafından böyle bir oyun gerçek insanlara denetildiğinde sonuçların gerçekten de 50, 25 ve 12 etrafında yoğunlaştığını görmüşlerdir. Kazanan seçim 13'e yakındı, ki bu da oyuncuların yaklaşık yüzde otuzunun seçtiği bir sayı olduğu görülmüştür. Bu oyunda en iyi strateji Nash dengesi olmasa da ondan çok da uzak değildir.

2.5. Denetlemek ve İzlemek

Başladığınız işi bitirmek çok zor bir süreçtir. Denetimi elinizden kaçırdığınız anda kendinizi kaotik ortamın bir soytarısı olarak bulursunuz. Proje yönetimi işi yapan, iyi bir ekiple çalışmalıdır. İyi bir ekip ile çalışmak sadece bilgi ve beceride akıllı ve zeki uzmanlar ile çalışmak anlamına gelmemeli, aynı zamanda dürüst ve güvenilir kişilerle çalışmak anlamına da gelmelidir. "Ben başardım, ben olmasam yapamazlardı." yerine "Herkes iyiydi, o yüzden başardık." stratejisinin geliştirilip uygulanması gerekir. İşler kötü gittiğinde aynaya, işler iyi gittiğinde ise pencereden dışarıya bakılmalıdır. Oynadığınız rolün farkında iseniz yaptığınız işler zaten sizin adınıza konuşacaktır.

Denetleme ya da izleme ile birlikte gözlem yapma, sorgulama ve mukayese ile ihlal bulma ve olumsuz davranışların kaynaklarının doğru teşhis edilmesi gerekmektedir. Tehditler ölçülürse yönetilir. Bu nedenle tehditler sürekli sorgulanarak ve mukayese edilerek ölçülmelidir.

Denetlemede başarısızlık nedenleri

- Kararlı ve tutarlı bir amaç oluşturulmaz.
- Kısa vadeli karlara öncelik ve önem verilir.
- Riskleri izleme ve gözden geçirmeler olmaz.
- Planlamalarda hatalar yapılır.
- Başarısızlık kabul edilmek istenmez, eksiklerin olduğu fark edilemez.
- Çalışanlarda ayrımcılığa uğradım düşüncesi oluşturulur.
- Yönetimde çok sık değişiklikler yapılır, liderlik kavramı yok edilir.
- İşler rakamlarla yönetilir.
- Zararlara yönelik ödemeler şişirilir, aşırı masraflar gizlenir.
- Sürekli iyileştirme çalışmaları benimsenmez.
- Stratejiler oluşturmadan kalite iyileştirme çabalarına girişilir.

- Yönetim çalışanlarla iletişimi keser ya da hiç kurmaz.
- Tedarikçi riskleri ve yetenekleri göz ardı edilir.
- Ekip oluşturulmaz ya da önemsenmez.
- Tüm yetkiler belirli bir grubun eline geçer, çalışanlara inisiyatif verilmez ya da yetkiler paylaşılmaz.

Denetlemede karşılaşılan sorunlar

- Görev bilinci yok ya da yeterince gelişmemiş, eksik.
- Tecrübe ve uzmanlığa saygınlık yok .
- Aşırı bürokratik eğilim.
- Hatalar, belirsizlikler, değişimler ve gelişmeler umursanmaz.
- Değişim istenmez, değişimlere direnme başlar.
- Disiplin eksikliği.
- Çevresi ile iletişim kurmada sorunlar yaşama.
- İstatistik alışkanlık yok.
- Bilişim teknolojilerinden yeterli ölçüde yararlanmama.
- Sanal ortamda sohbet odaları, oyun oynama ve sosyal paylaşım sitelerinde dolaşma zaafını kontrol edememe.
- Denetim ve teftiş korkusunu takıntı haline getirme.
- Kendini iyileştirmede yetersizlikler.
- İç müşterilerde memnuniyetsizliğin dayanılmaz boyutta olması.

2.6. Geri Dönüşü Yönetmek

Proje bitirildikten sonra, süreçleri izlemek, aksaklıkları bulmak; çözümler geliştirmek, üretmek ve uygulamak gerekir.

Bununla ilgili aşağıdaki örnek,
“Adım Danny Troatman. New Jersey’de yaşıyorum. Eşim ve çocuklarımla her akşam film seyretmeden önce şehir merkezinde bulunan markete dondurma almaya gidiyorum. Bir ay önce aldığım Porsche marka arabamla tabii ki… Fakat ne ilginçtir, ne zaman çikolatalı veya meyveli dondurma alıp arabama dönsem, araç çalışmıyor. Oysa vanilyalı aldığım zaman aracım rahatlıkla çalışıyor. Bunu bir kaç kere denedim ve her seferinde aynı sonucu aldım.Yardımlarınız için şimdiden teşekkürler.”

Porsche firmasındaki yetkililer derhal adı geçen bölgeye bir mühendis gönderiyorlar ve sebebini öğreninceye kadar orada kalmasını sağlıyorlar. Ertesi gün mühendis NewJersey’e varıyor ve Bay Troatman’la hemen temasa geçiyor. Aynı akşamdan başlamak üzere her akşam üstü mühendisimiz ve Bay Troatman dondurma almak üzere markete gidiyorlar. Gerçekten de çikolatalı ve meyveli dondurma alındığı zaman araba çalışmıyor, vanilyalı alındığı zaman ise rahatlıkla çalışıyor. Mühendis başlangıçta bu olaya şaşkınlıkla bakıyor fakat bilimsellikten uzaklaşmamaya gayret ediyor. Aradan yaklaşık bir ay geçiyor. Bay Troatman ile her gün markete giden mühendis, sonunda olayı çözüyor.

Yeni model Porsche arabalarda kullanılan soğutma sistemi, araç durdurulduktan hemen sonra devreye giriyor ve motor belirli bir ısıya düşene kadar motoru kilitliyor. Markette en çok satılan dondurma ise vanilyalı. Bu yüzden vanilyalı dondurma tezgahı önünde sürekli sıra oluyor. Bay Troatman sıraya girip dondurmasını alana kadar geçen süre,motorun soğuması için yeterli oluyor. Fakat çikolatalı veya meyveli dondurma tezgahı önünde sıra olmadığı için dondurmayı hemen alıp aracına geri dönüyor. Motor ise kilitli olduğu için araç çalışmıyor. Mühendis,raporunu yönetime sunuyor. Piyasadaki araçlar geri toplanıp, gerekli ayarlamalar yapılıyor ve müşterilere yeni haliyle teslim ediliyor.

2.7. Öğrenen Organizmaların Oluşturduğu Bünye (Ekip)

Bir organizmanın, değişimin izlerindeki ***tepki ve sapmaları takip ederek algılama yeteneğini geliştirmesi gerekmektedir.*** Bu aşamada belirsizliklerin sayısı oldukça fazladır.

Değişiklikler sınıflandırılırken doğruluğu artırmada eksik bilginin fark edilmesi için ***araştırma yapmaya yönelik akıl*** geliştirilmelidir. Bir yığın içerisinden aradığı bilgiyi bulabilmesi için diğerorganizamalr ile ***ekip olmayı becermeyi*** öğrenmelidir.

Ekibin bireyleri bilgi yığınını düzenleyen, bilgileri transfer eden tüm kontrol ve yönetim yazılımlarıdır. Ekip olmayı becerebilmek, ***problem çözmeye odaklı aklın gelişmesini*** sağlar.

Birlikte bilgi yığınları içerisinde dolaşan organizmalar fark ettikleri değişimleri diğerleri ile paylaşırken iş bölümü yapılmasını öğrenerek **süreç yönetmeye yönelik aklı** geliştirirler.

Daha büyük hedefi avlamada (tehditler) iş bölümü yapacaklarını planlamaya başladıklarında ise ***problem çözmeye yönelik katılımcı bir akıl geliştirirler***. Görev paylaşımında organizmalar üstlendikleri görevde başarılı olmaya ve en iyisini yapmada uzmanlaşarak organ gibi davranmaya başlarlar. Böylece ***problem çözmeye yönelik işin fonksiyonlarının paylaşımı*** ile organlar meydana getirilmiş olur. Organların birlikte hareket etmesinden bütünlük yani bünye meydana gelir. Bünyeyi meydana getiren organların birbirlerini hissetmeleri, görev paylaşımları, izleme, yönetme fonksiyonlarını yerine getirmesi için ***lider beyin oluşturmaları gerekmektedir***. Başarıya giden yolda ekip olma ve ekiplerin birbirlerini algılamaları ***hedefe yönelik katılımcı akıl*** ile mümkündür. Fırsatları yakalamada ve farklı olmada ya da farklılığı bulmada başarılı olmanın temel kuralı, takım olarak mükemmelliği gerçekleştirmeyi öğrendiklerinde ***kalite gücünü fark eden aklın geliştirilmesidir. Sadece tehditleri değil fırsatları*** yakalamada da farklı olmak gerektiğini hisseden organizmalar aynı anda tek bir noktaya odaklanabilmelidir.

Kendine ait bilgileri hızlıca toplayan, bütünleştirerek bünye meydana getiren organizmalar, bilgilerin kaynağı olan algılayıcıların davranışlarını izleyerek verecekleri tepkileri önceden kestirim yapmaya başlarlar. Tepkinin farklılığı ve şiddetini hızlı algılamak için sinir ağına benzer bir kontrol mekanizmasının kurulması gerekmektedir.

Yazılımsal algılayıcıların organizma gibi davranarak dost düşman ayrımı yapması, felaketleri, tehditleri ve saldırıları önceden algılaması gerekmektedir. Felaketlere ya da saldırıya maruz kalan veri yığını, o ana kadar sahip olduğu bilgileri korumak için kara kutuya sahip olmalıdır. Komşuluk ilişkilerinde çalışan algılayıcılar ile ön bilgi toplama, ara bilgi toplama ve merkezi bilgi toplama birimlerinde bilginin bozucu etkilerden korunması gerekmektedir.

Veri bütünleştirme, bilgileri kaynaştıran ve bünyeye birleştiren algoritmalardır. Hedeflerin davranışlarını bulmada, tanımlamada, takip etmede gerekli olan bilgileri toplayan ve sentez yapmayı öğrenen veri füzyonu algoritmaları da kullanılmaktadır. Bilgileri birleştirme işlevi, başta insanlar olmak üzere canlıların her zaman farkında olmadan yaptıkları sürekli bir işlemdir. Bir hareketin davranışının nedenini ve vereceği tepkileri kestirebilmek için toplanan verilerden yaşayan bir organizma oluşturulması gerekir. Öğrenen algoritmalar ile sürekli kendini geliştirerek yaşayan organizmanın tepkisel davranışını doğru kestirebilmek için diğer algılayıcılardan gelen bilgilerin organizma ile bütünleştirilmesi gerekmektedir. Örneğin tehditlere ait hedeflerin oluşturduğu bilinmeyen sayıdaki izler, toplanan bilgilerin bütünleştirilmesi ile hedeflerin yerleri ve davranışlarını belirleyebilir. Dağınık noktalara yerleştirilmiş çok sayıdaki algılayıcılardan gelen bilgiler hem çok karmaşık hem de çok fazla çeşit içerdiklerinden dolayı, toplanan bilgiler analiz edilirken karmaşık algoritmalar ve paralel işlemciler kullanılır.

Bilgiler kaynak potasında birleştirilirken;

- İz birleştirme
- Gözlem sentezi
- Kaynak yapma
- Birleştirme
- Tamamlama
- Toplama için gerekli algılama yöntemleri kullanılmalıdır.

Gördüğü nesneleri tanımak ve anlamlandırmak için öğrenen insanoğlu, kıyaslama yaparak farklılıkları ve değişiklikleri de bulur. Öğrenme süreklidir. Yaşam devam ettikçe öğrenme de devam eder. Öğrenilenlerin kayıt edildiği zihinsel bellek, düşünsel ve davranışsal değişimleri oluşturan bakış açısını belirler. İşin nasıl yapıldığını öğrenerek gerçekleştirme yeteneği kazanan birey, tecrübe ve deneyim kazanarak anlama ve kavramlaştırma yeteneği de geliştirir. Tepkisel davranışın nedeni bakış açısında gizlidir.

Davranışlar analiz edilirken eldeki veriler çoğu zaman yeterli olmaz. Hatta davranışlar doğru analiz edilmez ise yanlış yargılara da varılır. Covey: “Aynı enformasyona farklı bakış yargıyı belirler. ” diye özetler ve çözülemeyen sorunlar için zihin haritası ya da bakış açısını (paradigma) değiştirmenin gereğini vurgular. Einstein'in bir sözünde: “Karşılaşılan sorunlar, o sorunları meydana getiren düşünce ve davranış düzleminde kalarak çözülemez. ” der. Sorunların içinde kaybolmak yerine bakış açısını değiştirmeyi başarıp sorunlara farklı biçimde yaklaşıldığında çözme şansı da yakalanmış olur. “Başımıza gelen her şeyle onlara verdiğimiz tepki ve yanıt arasında geniş bir hareket alanı vardır. ” der Stephen Covey. Bu

nedenle yanıt ve tepki analizi yapılırken davranışların geçmişsel tecrübeleri ile programlanmış olduğunu varsayabiliriz.

Bünye kendisiyle aynı düzeyde olanlara bakarak onların sahip oldukları şeylere kendisinin de sahip olmasını ister. Sahip değilse bünye başkalarına göre bazı şeylerden mahrumdur. Bu mahrumiyet bünyenin karar vermesinde iç çatışma ile sonuçlanır.

Bünye, tepkisini ortaya koyarken içinde bulunduğu ortamın değerlerini dikkate alır. Bazen bünyenin değerleriyle çevrenin ya da ortamın değerleri çatışabilir. Bünye beklenti ve amaçlarını sınırlandırmak zorunda kalabilir. İstemeyerek yaptığı davranışlarda kendi içinde tutarlı olmak için haklı nedenler arar. Bütün bunları yapmaya çalışırken bünye zorlanır, sapma içinde olabilir.

Bünye rolüne uygun davranamadığı; amacını açık olarak belirleyemediği veya amacına ulaşmaya çalışırken engellendiğinde; kendisiyle aynı düzeydekilerle karşılaştırıldığında durumunu onlardan daha geri hissediyorsa birtakım sapma ve tepkilerle bunu dışarıya yansıtır.

Ekipler arası ilişkiler karmaşıklaştıkça herbir bünyenin amaç ve rol çatışmasını yaşamaları ihtimali sürekli artmaktadır.

Bünye, davranışı kestirmeyi öğrenerek, fark edip sorgulamaya başlar. Ayırt ederek fark bulur, iz arar, sezer, kuşkulanır. Tepkisel davranış geliştirir. Tahrik, kışkırtma, propaganda gibi tehditlerinde modellenmesi gerekmektedir.

Kendine ait bilgileri hızlıca toplayıp bütünleştirerek bünye meydana getiren organizmalar, bilgilerin kaynağı olan algılayıcıların davranışlarını izleyerek verecekleri tepkileri önceden kestirim yapmaya başlarlar.

Yapılanma karmaşıklaştıkça iç çatışmanın arttığı görülmektedir. Etkinlik ve verimliliklerin artırılarak hedeflenen amaca ulaşmayı gerçekleştirirken veri tabanı ve veri yığının davranışını izleyen ve yöneten yazılımların işbirliği içinde olması gerekir.

Rekabet durumunda taraflar kendileri için en iyisini yapmaya çalışırken aynı zamanda başkalarından da daha iyi yapmaya çalışırlar. **İşbirliği,** bir iş veya faaliyetle ilgili olarak müşterek hareket etmektir.

İkilem, iki şıktan birini istemeden seçme mecburiyetidir. İki alternatifin de eşdeğerde oluşu seçim yapmayı güçleştirmektedir. Alternatiflerden birisi daha cazipse seçim yapmak kolaylaşmaktadır. En önemli ikilem;" Kişisel çıkarlar mı yoksa genelin çıkarları mı daha önemlidir?" sorusunun yanıtıdır.

Karşı tarafın stratejilerini dikkate alıp ona göre davranma ilkesi, tarafların stratejilerinin mantığını anlama ve mantıksal kıyaslama yoluyla en uygun stratejiyi bulmaya yardımcı olur. Stratejiler seçilirken rasyonel davranmak esastır. İyi tanımlanmış bir çatışma durumunu ifade etmektedir. Kıyaslamada ilgiler genellikle çatışma halindedir. Rekabete dayalı ikilem yaşanırken kıyaslama ile sonuca gidildiğinde ya iki taraf da hiç bir şey kazanmamaktadır, ya da taraflardan birinin kaybı diğerinin ise kazancı söz konusudur.

İlgi ve çıkarlar kısmen çatışıyor kısmen uyuşuyorsa, tamamen kazanma veya tamamen kaybetme sonucu ortaya çıkmaz. İlgi ve çıkarların kısmen uyuştuğu kısmen de çatıştığı durumlarda ikilem içinde kalınmaktadır. Ortak çıkarlar mı yoksa kendi çıkarı mı tercih edilecektir? Kıyaslamada alternatif stratejiler sunulması ve hangi stratejinin hangi sonuca götüreceğinin gösterilmesi, ilgileri kısmen çatışan kısmen de örtüşen tarafların durumu diye belirtilebilir

Kıyaslamada rasyonellik ön plana çıkarsa ortama göre karlı sonuca götürdüğü söylenebilir.

Kişilerarası ilişkilerde insanlar üç şekilde davranır:

- İşbirlikçi (cooperative): Hem kendi hem de diğerlerinin çıkarlarını düşünür. Bu tür ilişkide karşılıklı güven ve arkadaşlık vardır.
- Bireysel Davranan (individualistic): Diğerleriyle ilgilenmeden kendi çıkarını en üst düzeye çıkarmaya çalışır. Diğerlerinin kayıp ya da kazancı önemli değildir.
- Rekabetçi (competitive): Kendi çıkarını düşünür ve aynı zamanda diğerlerinden daha iyi yapmaya çalışır.

Rekabet oyununda taraflar bazı silahlara sahiplerse işbirliği yerine engelleme eğilimindedirler. Etkili bir üçüncü kişi anlaşmaları için baskı yaparsa anlaşmak kolaylaşmaktadır. Yine çatışma küçük boyutluysa anlaşmak kolay olmaktadır.

Kişilerin her gün yaşadığı ve çatışmayla sonuçlanan faktörler ve süreçler şunlardır:

- Birbiriyle rekabet halindeki çok sayıda ihtiyaç ve roller,
- Dürtü ve rollerin ifade edilebileceği çok sayıda yol,
- Amaca ulaşmayı engelleyen pek çok faktörün varlığı,
- Ulaşılmak istenen amacın pozitif ve negatif yönleri.

Üç tür çatışma durumundan bahsedilebilir:

- Kişi iki sevdiği şeyde kararsız;
- Kişi iki sevmediği şeyde kararsız;
- Sevilen bir şeye yaklaşma korkusu .

Oluşturulan bünyede tepkiyi belirleyen pek çok faktör vardır. Organizasyonun kendisinden beklenen davranışları yerine getirebilmesi için bu rolleri öğrenmesi gerekir. Öğrenme sürecinde, roller de kendi aralarında etkileşim halinde olacaktır. Roller uyuşmuyorsa veya roller arası ilişki dengelenememişse rol çatışması söz konusudur.

- Çatışma veri yığını içerisindeki rol ve pozisyonundan kaynaklanır.
- İç çatışmayla başa çıkma teknik ve araçları her veri yığını için farklıdır.
- Yanlış entegrasyon ve tutarsızlıkların sebep olduğu iç çatışmayla başa çıkmanın yolu, rolleri değiştirmektir.

Çatışmaya sebep olan dört önemli faktörden bahsetmemiz uygun olacaktır. Bu faktörler: Amaç çatışması, rol çatışması, hüsran (frustration), ve rölatif mahrumiyettir.

Amaç Çatışması:

Üç tür amaç çatışması vardır:

- Yaklaşma-yaklaşma (Approach-approach): İki amaç arasından seçim yapma durumudur. Eğer iki amaç eşit değerde ise seçim yapmak daha da zordur. Amaçlardan birisinin değeri yüksek ise çatışma kolay çözülmektedir.
- Yaklaşma-sakınma (Approach-avoidence): Bir pozitif bir de negatif amacın çatışması halidir. Çözümü en zor çatışma türü olup rekabet-işbirliği ikilemi açısından önemlidir.
- Sakınma-sakınma (Avoidence-avoidence): İki negatif amaç arasından tercih yapma durumudur. Çözümü daha kolay bir çatışma türü olup örgütsel açıdan çok önemli değildir.

Her zaman örgütsel amaçlarla bireysel amaçlar örtüşmeyeceği için bu tür çatışmalar çok sık yaşanır..

Rol Çatışması:

Çok sayıda rol icra edildiğinden ve her rol için beklentiler farklı olduğundan sık sık rol çatışmaları görülür. Örgütsel açıdan bakıldığında, rol çatışmalarının olduğunu kabul ederek çatışmanın sebeplerini ortaya koymak sağlıklı bir davranış olur. Veri yığını içerisinde kendini var eden bünye, takımdaki rolü mü yoksa asli görevindeki rolü mü önemsemesi gerektiği noktasında karar verip uygun davranış göstermekte zorlanır. Bu iki ortamdaki rol farklılığı çatışmaya sebep olur.

Hüsran:

Hüsran (Frustration) : Amaca ulaşılmadan önce amaç bir sapma ile engellenirse hüsran oluşur. Pek çok savunma mekanizması olmasına rağmen hüsrana uğrayan bünye daha çok dört tür savunma mekanizması kullanır. Bu mekanizmalar: Saldırganlık, geri çekilme, saplanma ve taviz vermedir.

Hüsran Savunma Mekanizmaları

- Saldırganlık
- Geri Çekilme
- Saplanma
- Taviz Verme

Rölatif Mahrumiyet:

Rölatif Mahrumiyet (Relative Deprivation): Bünyeden beklenen verime ulaşamama duygusudur.

Stratejik Davranış Geliştirme:

Dinlemeye hazır olan ile konuşur.
İnsanlar ile karşılaştığında saygı ve alçak gönüllülük görür.
Ulaşılmaz görünmez.
İnsanlar ondan büyülenir.
Karşısındaki insanın neye ihtiyaç duyduğunu içinde hissedebilir.
İdrak eder, yani bilinen bir şeyi kavrar, nedenleriyle ve sonuçlarıyla anlar, bilinen şeyi özümser.
Ben bilen kişiyim demez. Düşünmeyi dene der. Cesaret verir.
Prensiplerini insanların anlıyabileceği tarzda konuşur ve bir hikâye veya misallerle izah eder.
İnsanlara karşı saygı gösterir, ayrımcılık yapmaz.

3. Proje Süreçleri Yönetimi

Süreç, işlevlerin planlanıp değer ya da ürün haline getirildiği, önceden belirlenmiş adımlardan oluşan iş akışıdır.Süreçlerde geliştirilen modeller, yazılım geliştirme sürecinin yapısını ve adımlarını belirler. Planlanmış bir süreç, zamanında ve kaliteli bir ürün elde edilmesini sağlar.
Malın, hizmetin, fikrin, kültürün veya bir ürünün üretilmesine değer denir. Üretilecek ürünün, hizmetin, fikrin ya da kültürün istenen nicelikte, istenen zamanda ve istenen nitelikte yapılması zorunluluğunu üzerine alan kişi ya da gruplardan oluşan paydaşlar ! süreci planlamalı, yapılışını ve niteliklerini denetlemeli, yöntemlerini irdelemeli, işletme düzenini ve malzeme akışını kontrol etmelidir.

Süreç adımları:

- Düşüncelerin toplanması, düşüncelerin ayıklanması, ürün kavramı oluşturulması, ürün geliştirme, pazara hazırlama, sınama, pazara sunuş.
- Yeni ürün kaynakları: müşteri, satıcılar, çalışanlar ve aileleri, yöneticiler, rakipler, krizler, sıkıntılara çare olma.
- Yeni fikirler hangi durumlarda ortaya çıkmaktadır? Yeni fikirleri hızlandırıcı etkiler. 2. Dünya Savaşı, uzay, beyin.
- Kim (sosyal), ne (fonksiyon), nerede (konum), ne zaman, nasıl (duygusal), neden (anlamsal)
- İhtiyaç, problem ve problemlerin araştırılması.
- Senaryo: soruna nasıl çözüm olmalıdır?
- Huavey: Cisco Network ürünlerini kopyaladı, sonra farkındalık yarattılar, son kullanıcıların dertlerini fark ettiklerinde çözüm ürettiler.

Paydaşlar, etkileyen ve etkilenen menfaat sahipleridir; *çalışanlar, yatırımcılar, kurucu sermayedarlar, hissedarlar, müşteriler, tedarikçiler, iş ortakları, kamu kurum ve kuruluşları, bankalar ve finans kuruluşları, sendikalar, danışmanlar gibi tekneden ekmek yiyecek olanlardır.* Faaliyetleri etkin bir şekilde gerçekleştirirken, engellerin belirlenip giderilmesi için stratejiler geliştirmede paydaşların birbirleriyle olan ilişkileri ve olası çıkar çatışmaları doğru tespit edilmelidir. Kuruluşun güçlü ve zayıf yönleri hakkında paydaşlardan fikir edinilmesi, paydaşların hangi aşamada katkı sağlayacağının tespit edilmesi, paydaşların görüş, öneri ve beklentilerinin stratejik planlama sürecine dahil edilmesi, planın bu kesimlerce sahiplenilmesi başarılı olma şansını artıracaktır.

Faaliyetlerin yerine getirilmesinde katkısı olanlar, yönlendirenler, ürünlerini ya da hizmetlerini kullananlar, etkilenenler, etkileyenler, kısaca sınıflandırılmış beklentileri olanlar ile ilişkiler dürüstlük temelinde doğru kurulmalıdır. Paydaşların, işletmenin faaliyetlerini etkileme gücü ile

faaliyetlerinden etkilenme derecesi doğru analiz edilmelidir. Paydaş görüşleri alınmasına karar verildiğinde mülakat, anket, toplantı gibi uygulanacak yöntem belirlenirken görüşülecek kişi sayısı, erişilebilirlik, önem sırası ve etkisi gibi etkenler göz önünde bulundurulmalıdır. Kurum üzerindeki etkisi güçlü olan paydaşlar ile yüz yüze görüşme yapılması, bu kesimlerle olan iletişimin güçlendirilmesinde etkili olacaktır. Görüşülecek kişi sayısının yüksek olduğu durumlarda ise anket uygulanması daha uygun olabilir. Paydaşların hangi faaliyetleri ve hizmetleri önemsedikleri, geliştirilmesi gereken uygulamalar ve beklentiler gibi konularda detaylı görüşlerinin alınması gerekir.

3.1. Planlama ve Analiz

Planlama, iş akış süreçlerini tanımlar, disipline eder. Riskleri oluşturan değişimlerin, belirsizliklerin krize dönüşmeden nasıl bulunacağını tanımlar. Her ne sebepten olursa olsun kriz ya da kaos ile karşılaşıldığında yok oluşa sürüklenmeden var olmaya dönüşümün kurallarını işaret eder. Stratejik planlama ise hedefe ***yönlendirmeyi*** tanımlar.İş süreçlerinde, iç ve dış rakiplere ilişkin fırsatların ve

tehditlerin incelenip güçlü ve zayıf yönlerin kıyaslanarak belirlenmesi stratejik planlama ile mümkündür. Mal, hizmet, fikir ya da kültür olarak üretilecek değerlerin hammaddeden üretime, satılmasından müşteri memnuniyetinin izlenmesine kadar tanımlanan işlere iş süreçleri denir. İş süreçlerinde planlanan aktivitelerin performanslarını izlemek, gözden geçirmek, potansiyel riskleri bulmak ve etkilerini düşürerek tekrarlanma olasılığını azaltmak için yapılanlar ise iş süreç yönetimi olarak adlandırılır.

Stratejik planlama geliştirilirken; süreçlere ve elde edilecek sonuçlara ***odaklanma*** sağlanmalıdır. Süreçler düzenli olarak gözden geçirilmeli, ***değişimler aranarak*** riskler bulunmalıdır. Gerçekleşebilecek ***gerçekçi*** bir gelecek resmedilmelidir. Kurum kendisini tanımlarken, neyi niçin yaptığını değerlendirirken, şekillendirirken, temel kararları ve eylemleri üretirken, kaliteli yönetim sürecinde ***disiplini*** önemsemelidir. İzlenmede, değerlendirilmede ve denetlenmede ***hesap verme sorumluluğu*** oluşturulmalıdır. Tüm paydaşların ortak çaba ve desteği ile ***katılımcı bir yaklaşım*** sergilenmelidir. Kurumun yapı ve ihtiyaçlarına göre uyum sağlayan ***şablon değil esnek bir yönetim*** anlayışı oluşturulmalıdır.

Stratejik planlama, kurumun dört temel soruyu cevaplandırmasına yardımcı olur:

- Durum analizi ile ***"Neredeyiz?"*** sorusuna yanıt aranır.
- ***"Nereye gitmek istiyoruz?"*** sorusuna yanıt ile kurumun hedefi belirlenir.
- Hedefe ulaşmak için kullanılacak yöntemler için "**Gitmek istediğimiz yere nasıl ulaşacağız?"** sorusu yanıtlanır.
- Değerlendirme sürecinde ***"Başarımızı nasıl takip eder ve değerlendiririz?"*** sorusu yanıtlanır.

Stratejik planlama; kurumsal kültür ve kimliğin gelişimine ve güçlendirilmesine destek olur.

3.2. Proje Geliştirme

Müşteri odağı hep sürecin içerisinde olmalıdır. Stage – Gate yöntemi, süreçlerin belirlenen adımlarında analiz edilmesini, kontrol edilmesini ve sorgulanmasını gerekli kılar.Fikirden başlayarak bütün adımlar takip edilir. Denetçi test eder, geçişe izin verir ya da vermez.

Bilgi birikimi ve bilinen teknolojilerden boşluk var ise yapılabilecek ürünü geliştirmek, üretmek ve satmak hedeflenir. Bilmediğiniz teknoloji ile inovasyon olmaz. Öğrenmek gerekir. Sürece dahil olan herkes toplantıya dahil olmalıdır. (Patron, asistan, ArGe, tasarım, üretim, satış, ...)

Araştırmada ürün tanımlanır. Ürünün karlılık analizi yapılmaldır. Nasıl satılacak, ürünün detayları ne olacak?

Zaman planlamada kritik adım ve esnek adım tanımlanmalıdır.

Paralel süreçlerde uzun süreçli adım ile birlikte ilerleyen kısa süreçli işlerin yürütülmesinde esnek davranış geliştirilebilir.

Stage - Gate yönteminin faydaları:
Kontrol Kapılarında teknoloji, pazar, hedef, konum sürekli sorgulanır. Süreci yürüten bütün ekip baştan sona işin içerisindedir.
Bilgi birikimi elde edilir.
Direnç gösterenler, uyum göstermeyenler ortaya çıkar.
Herşey alenidir, ortadır.
Bir süreç baştan sona takip edilir.
Zaman doğru planlanır. Etkin kullanılır.
Doğru sonuç garantilenir. Geri dönüş engellenir.
Ekonomik kazanç odaklıdır.
Çalışanların motivasyonu artırılır, fikirleri önemsendiğinden değerli hissi verilir.
Uyumu artırır. Ortak akıl oluşturur. Organizasyon hizalanır.
Ürün fikrinden Pazar bilinir.
Kaynaklardaki israflar azaltılır.
Ara çıktılara odaklanıılır. Zayıf projeler erken ölür.

3.3. Tasarımda Etkileşimlerin Yönetilmesi

Lineer düşünmede engel olarak karşımıza çıkan duvarı nasıl aşarız?
Problemlere çözüm aranırken ya da innovatif bir düşünce yaratılırken; neden sonuç ilişkisi önemsenir. Yaratıcılık yetenek değil, düşünme ve davranma biçimidir. Üretirken kapalı modda sonucu belli iş yapılır, stres vardır, mizah yoktur, ikna yoktur. Açık modda ise mizah vardır, ikna vardır, mekan özgürdür, zaman sınırsızdır, ısrar etmekten ve hata yapmaktan korkulmaz. Yaratıcılık oyundur, oyunda zaman, mekan ve kurallar için belirli sınırlamalar olamaz.

Engel ile karşılaştığında, özellikle bu engel aşamayacağın bir duvar ya da hendek ise etrafı dolanılır. Kişilerin sorunlara farklı yönlerden bakabilmeyi ve geniş düşünebilmeyi öğrenmelerini sağlamayı amaçlayan düşünme biçimidir. Lateral düşünce, klasik düşünce kalıplarının dışına çıkmaktır. Konu objektif, olumlu, olumsuz, duygusal, yaratıcı, değerlendirici açılardan düşünülür. Amaç kişilerin sorunlara farklı yönlerden bakabilmeyi ve geniş düşünebilmeyi öğrenmelerini sağlamaktır.

En bilinen örneklerden biri şöyledir: Bir ormanın derinliklerinde üzerinde sadece mayo ve snorkel olan bir adam ölü olarak bulunmuştur. En yakın göl yaklaşık 8 km uzaklıktadır ve denize olan mesafe de yaklaşık 50 km'dir. Adam nasıl ölmüştür? Orman yangını sırasında bir itfaiye uçağı yakındaki gölden su alıp yangının üzerine dökmüştür. Uçak tesadüfen o sırada gölde yüzmekte olan adamı da alıp ormana atmıştır. Konuya bambaşka bir yönden bakabilme yeteneğini geliştiren alıştırmalardır. Alexander Fleming bakteri üreten kapları incelerken kabın birinde bakteri olmadığını görmüştür. Yani hiçbir bakterinin üremediğini fark etmiş ve penisilini bulmuştur.

Reklamcılık ve metin yazarlığında kullanılan bir takım teknikler de bu gruba girer. Konuya tersinden bakma, başka birinin gözünden bakma, uzaklaşıp dışarıdan bakma gibi teknikler kullanılır. Çocukluktan başlayıp geliştirilen bir takım düşünme oyunları faydalı olabiliyor.
Önemli olan düşünce çeşitliliği yaratmaktır. Finansman tablolarından nakit akışlarına kadar başınızı kaşıyacak vaktiniz olmadığından şikâyet ediyorsunuz. Haklısınız. Hemen başınızı kaşıyacak birini bulun. Unutmayın ki her hastalığın tedavisini bilen bir uzman vardır bu dünyada. "Başkalarının hangi noktalarda yanıldığını özenle araştır. O zaman dikkatli olmanın ne demek olduğunu iyice öğrenmiş olacaksın!"

3.4. Yol Haritası ve Eylem Planı Hazırlama

Amaç,

- Dünden bugünü değil yarını hazırlamak.
- Ortak bir akıl oluşturabilmek.
- Yaratıcı problem çözüm yöntemleri geliştirmek ve yol haritası hazırlamak.
- Farklı süreç adımlarını biraraya getirip plan oluşturarak örtüştürmek.
- Yol gösterici raporlar hazırlamak
- Teknolojik gelişmelerin katkısıyla oluşturulacak değerler (hizmetler, ürünler, bilgelik, ...) ile pazar hedeflerinin örtüştürülmesi hedeflenir.
- Kaynakların organizasyonu ile hedef çıktılara nasıl ulaşılacağı planlanır.

İş süreçlerini gerçekleme aşamalarında, sorumlulukların zaman planlamasında detaylandırıldığı bir eylem planı hazırlanmalıdır. Sorumluluklar, planlamadan etütlere, hammaddeden üretime, üretimden satışa, kalite kontrolden müşteri memnuniyetine, iş görenlerden yönetime kadar bütünü oluşturan organizasyonun organizmalarını kapsar. Eylem planı, organizasyonun belirlenen stratejik amaç ve hedeflere ulaşmasını sağlamak üzere, iş süreçlerini gerçekleştirme aşamalarında hayata geçirilecek faaliyet ve projeleri içermektedir. Eylem planında öncelik,planın hedeflere paralel olacak şekilde düzenlenmesidir. Planlanan zaman dilimlerinde, gerçekleştirilecek eylemlerle ilgili açıklamalar, iş görenler ve iş süreçlerinin tarihleri ve sürelerine ilişkin bilgiler yer almaktadır. Eylem planı hazırlanırken ! iş ahlakına, çevre koruma bilincine, ilk yardım bilgisine, el becerisine ve mesleki bilgiye sahip iş görenlerin organize edilmesine öncelik verilmelidir.

İş güvenliği ve işçi sağlığına ilişkin eylem planları hazırlanırken; İş ve işçi güvenliğinde işe uygun iş elbisesi seçilmeli, ilk yardım malzemeleri bulundurulmalıdır. Çevre kirliliğini önleyici tedbirler alınmalıdır. Atıklar, geri dönüşümü olabilecek mesleki atıklar ve geri dönüşümü olmayan zararlı mesleki atıklar olarak sınıflandırılmalı ve depolanmalıdır.

İş disiplini içerisinde işyeri çalışma koşulları ve normlarına uygun çalışanlar, tutum ve davranışları ile dürüst, çalışkan, çevre korumaya karşı duyarlı ve dikkatli olmalıdırlar. Çalışma şekli ve saatleri organize edilmelidir. İş görenler detaylara özen göstermeli ve sorumluluk sahibi olmalıdırlar. Güler yüzlü olmalı, insan ilişkilerine özen göstermeli, sabırlı ve soğukkanlı olmalı, temiz olmaya özen göstermeli ve yeniliklere açık olmalıdırlar. Çalışanlar; araştırma, ekip içinde çalışma, el becerisi, ikna, iletişim, karar verme, öğretme, öğrenme, problem çözme, dinleme ve kayıt tutma gibi yeteneklere sahip olacak şekilde donatılmalıdırlar. Kaliteye dikkat etmeliler, kararlı, planlı ve organize olmalılar ve zamanı iyi kullanmalıdırlar.

Alan tespiti (Keşif) yapılırken alanın fiziki detaylarına dikkat edilmelidir. Projeye göre alanın enerji besleme noktaları belirlenmelidir. Haberleşme tesisatı çalışma prensiplerine göre

sistemin yerleşim planı doğrulanmalıdır. Alanın malzeme nakil noktaları belirlenmelidir. Malzeme hazırlığı yapılırken malzeme listesi incelenmeli, kaliteye dikkat edilerek malzeme temin edilmeli, listeye göre malzemelerin son kontrolü yapılmalıdır.

Riskler ve riske tepki yöntemleri ekip olma anlayışı geliştirilmelidir. Elektrik, su, gaz gibi tesisatlardan kaynaklanabilecek riskler sürekli gözden geçirilmeli, değişimler izlenmelidir. Çalışanlar risklerin neden olacağı tehlikelere karşı uyarılmalıdır. Risklere ve oluştuğunda krizlere karşı organizasyon yeteneği geliştirmek için çalışma ekibi içerisinde koordine sağlanmalı, yapılan işe göre güvenlik teçhizatları kullanılmalıdır. Sorunsuz çalışma ortamı, uygun aydınlatma koşulları sağlanmalıdır.

Dokümantasyonlar, servis formları, ürün teslim alma ya da ürün teslim etme belgeleri, sistem test raporları, garanti belgelerinin nasıl doldurulacağı çok iyi bilinmelidir. Sisteme ait diyagramlar ve sistem yapılandırmaları, haritalar, kanallar ve kablo güzergâhları, yer altı odaları gibi **alt yapıya** ilişkin projeler hazırlanmalıdır. Tedarik ve ihale dokümanları, yönetmelikler ve kanunlar, mesleğe ilişkin yasal düzenlemeler, yayınlar ve kataloglar temin edilmelidir.

Malzemelerin ve hizmetlerin, hazırlanan liste doğrultusunda, birim fiyatı esas alınarak **mali analiz** yapılmalı ve maliyet hakkında üst yönetim bilgilendirilmelidir. Planlama, malzeme kataloglarına göre kullanılacak araç gereç ve ekipmanlar, işlerin öncelik sıralamasına göre yapılmalıdır. Üst yönetim ikna edilerek doğru bir şekilde onayı alınmalıdır. Satın alınacak hizmetler tanımlanmalıdır. Aktivitelerin bütçeleri ve maliyetleri hesaplanmalı, finans döngü tabloları hazırlanmalıdır.

Lojistik ve nakliye işlerinde kent yerleşim planını kullanarak ulaşım güzergâhı hazırlanmalı, gerekli izinler alınmalıdır. Malzeme nakledilirken güvenlik tedbirleri alınmalı, malzemeler sistemli bir şekilde araçlara yüklenmeli, sevk irsaliyesi düzenlenmelidir.

Tahmini iş bitim süresi ve montaj değişiklikleri üst yönetime raporlanmalıdır. İş süreçleri gerçekleme aşamalarında içerikler, kaynaklar, aktiviteler arasındaki ilişkiler, programlar ve başlangıç-bitiş süre tahminleri tanımlanmalıdır. İş planı ve zaman çizelgesi, etkinlik süre tahminleri hazırlanmalıdır. Kaynak planlaması, lojistik, tedarik, sipariş, temin ve teslim tarihleri, sevk irsaliyesi düzenleme öğrenilmelidir.

Ekip sorumlulukları ve yetkilendirmeler çok iyi organize edilmelidir. Kontrol, izleme ve raporlamada kıstasların neler olacağı belirlenmelidir. İşyerine ait araç, gereç ve ekipmanlar itinalı kullanılmalıdır. İşçilik (maaş, sigorta, stopaj), ulaşım, konaklama ve yemek giderleri, vergiler gibi **giderler** detaylandırılmalıdır. **Eğitimler** düzenlenmelidir. **Müşteri ile etkili iletişim** yapılmalıdır. Tüm iş süreçlerine ilişkin **veri tabanı** hazırlanmalıdır. Yükleniciler,

taşeronlar, idare ya da işverenler, ekipman ve malzeme sağlayıcılar, danışmanlar ile **sözleşmeler** imzalanmalıdır.

Temel mesleki bilgiler öğrenilmelidir; Bilgisayar, elektronik, elektrik, elektrik tesisatı ve elektrik malzeme, ısı ve ışık, mesleki terimler; Fizik, kimya, matematik; Ölçme; Yabancı dil; Teknik resim; Standartlar; Hata ve alarm mesajları; Arızalar; Testler; Soğutma ve ısıtma sistemleri; Yedek malzeme; Topraklama; Paratoner. Atık depolama, çevre koruma ve düzenleme, ilk yardım, iş güvenliği önlemleri, işçi sağlığı ve iş güvenliği önlemleri gibi destek bilgilendirme önemsenmelidir.

3.5. İş Süreçleri Planlama

Mal, hizmet, fikir ya da kültür değerlerini üretmek için planlanan aktivitelerin performanslarını izlemek, potansiyel riskleri ve değişimleri bulmak ve etkilerini düşürerek tekrarlanma olasılığını azaltmak için yapılanlar iş süreçleri yönetimi olarak adlandırılır. İş süreçleri yönetiminde, değişimleri görmek için, oluşabilecek sorunlar ve sapmaların değişen değerleri izlenmeli ve sürekli analiz edilmelidir. Hedefe ilerlerken aşılması imkânsız engeller ile karşılaşıldığında rotadan sapmamada ısrar etmek ya da değişimlere direnmek yerine dönüşerek fırsatlara yönelmek önemsenmelidir.

İş süreçleri uygulama aşamasında değişimlere neden olan faktörler:

- Rekabetin detaylı analiz edilememesi,
- Kültürel farklılıkların ayrımcılığa dönüştürülmesi,
- Yönetim desteğinin olmaması,
- Hatalı tanımlanmış hedefler ve ihtiyaçlar,
- Katılımda eksiklikler,
- Yetersiz ve etkisiz planlama,
- Risk yönetiminin planlanmaması olarak sıralanır.

İş süreçleri uygulama aşamasında sapmalara neden olan faktörler:

- Maliyet planlamasında sapmalar,
- Süreçleri kontrol ve izleme aşamasında aksaklıklar,
- Ekip içindeki sorunlar,
- Fizibilite ve değerlendirme aşamasında unutulanlar,
- Enformasyon sistemindeki tıkanıklıklar,
- Kapasite ayarlamasında sorunlar,
- Ekonomik konjonktürdeki olumsuzluklar (özellikle enflasyon),
- Hedeflerin gerçekçi koyulmaması,
- Kısıtlı kaynaklar,
- Yetersiz stoklar,
- Bürokratik engeller olarak sıralanır.

İş süreçleri uygulama aşamasında karşılaşılacak değişimler ve sapmalar ile baş edebilmek için yaratıcı, kontrol ve denetim yetkisine sahip yönetimler oluşturulmalıdır. Ekibinden sorumlu olan lider, yöneticilikten çok yönlendirme yapmalı, ekibi motive etmeli ve onlara yol göstermelidir. Ekibindeki her üyenin yaptığı işe yönelik güncel bilgilerle donatılmalıdır. Eğitim ile sadece kişilerin gerekli ve yeterli bilgiye sahip olmalarının ötesinde aynı dilden konuşmalarını da sağlamalıdır.

İş süreçlerinin uygulama aşamasında sorun çıktığında, lider ile toplantı düzenlenmeli, beyin fırtınası yapılmalı, ortak akıl üretme temelinde herkes çekinmeden düşüncesini açıklamalıdır. Tüm ekibin görüş birliğinde çözümler üretmesi önemsenmelidir. Çalışanlar, bilgi paylaşımı ve sağlıklı iletişim kurmanın gerekliliğine inanmalıdır. Tecrübe ve deneyimlerin paylaşılması ekibe güç verecektir. Çalışanlar arası ilişkilerin hızlıca sıcak yakınlaşmaya dönüşmesi; sorgulamada ve araştırmada yanlış yanıtların alınmasına neden olacağı göz önünde bulundurulmalıdır. Ekip izlenmeli uyumsuzlar ve problem çıkaranlar ayıklanmalıdır. Cezada işten kovma veya kovulma birinci öncelik olmamalı, kişinin ders alması önemsenmelidir. Takdir edilmenin çalışanlara yüksek isteklendirme sağladığı unutulmamalıdır. Ekip yönetiminde görev listesi hazırlanmalı, kaynakların planlanması yapılmalı, faaliyetler izlenmelidir. Çalışanlar arasında müsaade edilecek çatışma ve yetkilendirme sınırlarına bağlı olarak iş tanımları doğru yapılmalıdır. *Çalışanlarda yönetimin desteği hissi oluşturulmazsa, heyecan kaybolur ve önemsenmediği duygusu oluşur.* Hedef belirsiz olursa izlenecek yol da yok olur.

Çevre ve iş güvenliği ile birlikte çalışanların yemek, ulaşım ve konaklama hizmetlerinin kaliteli ve ayrımcılık yapılmadan sağlanması gerekir. Kontrol, denetim ve izlemenin yanı sıra tüm süreçlere ilişkin toplanan bilgiler mukayese edilerek, analizler yapılarak ihlaller ve tehditler araştırılmalıdır. İhlaller ya da tehditler algılandığında erken uyarı ve acil müdahale süreçlerinde risk ve sonrasında kriz yönetimi başlatılmalıdır. İhlal, sapma, suç unsuru, adli delil bulma gibi fonksiyonlar iş akışının denetiminde stratejik öneme sahiptir. Kriz ve riskin daha oluşmadan fark edilebilmesi için raporlama ve istatistiksel veritabanı oluşturulması gerekir.

Maliyet, teknik performans ve iş zaman diliminde birlikte izlenmelidir. "Bitmesi gereken aktivitelerin şu anki durumu gelecekteki aktiviteleri nasıl etkiler?","Nasıl doğrular?","Nasıl yanlışlar?","İhtiyaçlar nelerdir?" sorularına yanıt aranırken iş programı geliştirilmelidir. Planlanan ve gerçekleşen aktivitelerin başlama/bitiş tarihleri, planlanan programdan sapmalar ve projeye etkileri belirlenmelidir. İş program yardımı ile aşamalardan gelen kalite göstergeleri takip edilmelidir. Planlanan işlerle gerçekleşenler maliyet, başlangıç ve bitiş tarihleriyle karşılaştırılmalıdır. Gerçekleşen harcamalardan projeyi tamamlamak için gerekli bütçe tahmini yapılmalıdır. Herhangi bir sapmanın diğer görevleri nasıl etkileyeceğine ilişkin öngörülerde bulunmak gerekir. Gecikme ya da plandan sapma olduğunda, bekleyen görevler gözden geçirilerek, süreçlerdeki diğer görevlerde nasıl bir etki yaratacağı belirlenmelidir. Ek ayarlamaların gerekip gerekmediğine karar verilmelidir.

İş süreçleri uygulama aşamasına geldiğinde, bilginin doğru toplanması sağlanmalıdır. Durum raporları, güncellemeler, finansal analizler, planlanan maliyetlerle gerçekleşenlerin ve bunlar

arasındaki farkların ölçümleri sürekli yapılmalıdır. İş süreçleri yönetiminde amaç, performans, zaman, maliyet ve kapsam bileşenlerinin izlenmesi ve risklerin önceden belirlenmesidir.

İşletme ya da kurum, bölümler yerine süreçler olarak tanımlanmalıdır. Süreç adımlarında istekleri belirlerken, beklentilere farkındalık katacak öneriler verilmelidir.
Süreç: girdileri çıktılara dönüştüren birbirleri ile etkileşimli (iyileştirme, sahipli, ölçülen, kıyaslanan) faaliyetler zinciridir. Süreçlerde sorgulama döngüsü oluştururken başarı hikayeleri önemsenmelidir.

Kurumsallaşmayı yok edici faktörler; üretkenliğin olumsuz etkilenmesi, hatanın ve yanlışın atlanması, farkına varılmaması.

Etkenlik, verimlilik, üretkenlik:
Verimlilik: Üretimde, üründe, süreçte, malzemede aynı işin yapılış maliyetini geri kazanmak, boşa giden enerjiyi kazanmak.

Etkenlik: Zamanında teslimat. Gerçeklenen, planlanan hizmet verme süresi, müşteri şikayet oranı.

Üretkenlik: Girdi ile çıktı arasındaki ilişkiyi gösterir.

Süreç performansının ölçülmesi:

- Proje geliştirme yönetiminde lider davranış geliştirmek.
- Başarı hissi oluşturulması
- Gelişmeyi gözleme
- Belirleyici olmak
- Hataların nedenlerini bulmak
- Hedefe olan mesafeyi ölçmek
- Daha sıkı sarılmak
- Miktar ve süre

3.6. İstatistiksel Süreç Analizi

Bir işletmede verilere dayalı, gerçekçi karar verilebilmesi ve sürekli iyileştirmenin sağlanabilmesi için öncelikle hataların temelinde yatan değişkenliğin belirlenmesi, azaltılması ve belli sınırlar dahilinde tutulması gerekir. Kalitenin temel kuramları olan ölçüm, analiz ve kıyaslama doğru yapılırsa hatalar önlenir, maliyetler düşer, verim artar. Tüm olayların temelinde değişkenlikler vardır. Hataların çok büyük bir bölümü de değişkenlikten kaynaklanır. İstatistik teknikler kullanılarak değişkenliğin özellikleri incelenmeli ve hataların kaynakları tespit edilmelidir. İstatistiksel süreç yönetiminde istatistik teknikler analize yardımcı olur, iletişimi kolaylaştırır, konuya farklı açılardan bakan kişilerin aynı dili konuşması sağlanır. İstatistik, iş süreçlerinin gözlenmesi ve bilgilerin sistematik olarak toplanması ve işlenmesi sonucunda belirli duyarlılıkta risklere yönelik tahminde bulunmayı ve yorum yapmayı sağlayan bilim dalıdır.

İş süreçlerinde, periyodik olarak, bulunulan noktadan ileriye doğru bakılarak bitimde beklenen maliyet öngörülmeye çalışılır. Her aktivitenin süre ve maliyetleri tahmin edilir. Gerçekleşme bilgileri geldikten sonra kalan iş ve buna bağlı maliyetler yeniden tahmin edilir. Bugüne kadar gerçekleşen maliyetler bulunur. Proje için kullanılabilecek ya da gereken fon miktarı belirlenir. Eğer maliyetler artacaksa o zaman; projede yeniden maliyet hesabı yapılır, geri kalan maliyetler eldeki fonlarla karşılanana kadar gereksiz ve önemsiz gereksinimler elimine edilir. İdari yönetime, gelinen noktadaki proje maliyetinin tahmin edilenden daha fazla olacağı ve gerekli kaynakların yaratılması gerektiği rapor edilmelidir.

İstatistiksel süreç kontrol yönetimi ile istatistiksel teknikler kullanılarak kalitenin korunması ve geliştirilmesi amaçlanmaktadır. Temel olarak sıfır hata ile ürün üretmek pratikte mümkün olmasa bile hata oranının olabildiğince azaltılması istenmektedir. Çünkü tek bir tüketicinin bile memnuniyetsizliği, ürünün pazardaki şansını olumsuz etkileyecektir. İstatistiksel kalite kontrol; en az maliyetle, zamanında ve doğru veri üretmektir.

İstatistiksel kalite kontrol teknikleri hata büyüklüklerinin kontrolüne yardımcı olmaktadır. İstatistiksel kalite kontrol metotları süreç kontrolü ve ürün kontrolü olarak ikiye ayrılır. Üretimin, önceden belirlenmiş kalite teknik özelliklerine uygunluğunu sağlayan, standartlara bağlılığı hedef alan, uygun olmayan ürün üretimini en aza indirgemekte kullanılan bir araçtır. Böylece düzeltici ve önleyici faaliyetlerin başlatılabilmesi için verilere dayalı karar verme olanağı sağlar. İstatistiksel süreç kontrolünde Dr. E.Deming'in yorumuna göre; kaliteyi yükseltmek maliyetleri düşürür. Kaliteyi yükseltmenin yolu, hataları önlemektir. Muayene ile kaliteyi sağlamak zor ve pahalı bir işlemdir. Kalitesizliğin temelinde değişkenlik yatar. Kalite birdenbire sağlanamaz, sürekli gelişme ile istenilen düzeye ulaşılabilir. Sistem süreçlerinin kontrol altına alınması ancak istatistiksel süreç kontrolü ile mümkün olur. Makine, takım, yöntem, malzeme, operatör, bakım ve çevre şartlarından kaynaklanan bütün süreçler değişime uğrarlar. Hiçbir zaman iki ürün veya ürünün herhangi bir özelliği aynı olamaz. İşlenen parçaların ölçüleri ve özellikleri arasında küçük de olsa mutlaka birbirine göre fark vardır. Bu durum teknik özelliklerin niçin toleransları olduğunu açıklar. Birçok küçük kaynaktan

oluşan ve her süreçte tesadüfi olarak her an değişik seviyelerde bulunan değişimler önceden tahmin edilebilir. Ancak, değişimlerin tespit edilmesi ve düzeltilmesi zordur. Bununla birlikte süreçteki belirgin riskli değişkenler elimine edildikten sonra, zamanla riski az olan ama kararlı bir değişim ya da dağılım gösterenlerin azaltılması yoluna gidilmelidir. Değişimlerin sebepleri belirsiz bir kaynaktan oluşur, önceden tahmin edilemez ve düzenli değildir. Önlem almadıkça tekrar ederler. Değişimlerin ne zaman ortaya çıktığı bilinirse kolaylıkla tespit edilebilir ve düzeltilebilir. Amaç değişimin sebeplerini ortadan kaldırarak süreçleri kontrol altında tutmaktır. Kontrol altındaki bir süreç, değişimin sebepleri izlendiğinden ve ortadan kaldırıldığından sürekli olarak kendi doğal toleransları içinde ürünler üretir. Süreç istatistiksel olarak kontrol altında ve sürekli olarak kendi doğal toleransı içinde ürünler üretiyor ise sürecin yeterliliğinin belirlenmesi için doğal toleranslar teknik özelliklerinin toleransları ile karşılaştırılmalıdır. İstatistiksel süreç kontrolü, problemleri önceden belirlemeye imkan sağlar, üründeki değişkenlikler azalır, ürün kalitesi gelişir, hurda oranı azalır, etkin kapasite kullanımı artar, birim maliyet düşer, kontrol faaliyetleri azalır, kalitesizlik maliyetleri düşer, makine veya süreç yeterliliğinin izlenmesine imkan sağlar, düzeltici ve önleyici faaliyet ihtiyaçlarını belirler. Problem çözülünce sürecin istatistiksel olarak kontrol altında olduğunu görmek için analize devam edilmelidir.

Kök Neden Analizi:

Suçlu aranmaz. Tarafsız – önyargısız, analitik yaklaşım geliştirilmelidir.
Problemin niçin ve nasıl meydana geldiği araştırılmalıdır.
Aksaklıkların düzeltilmesinde maskeleme önemsenmelidir. Maskeleme problemin kronikleşmesine neden olur, kalıcı çözümler üretilmelidir.
Gerçekleşme
Sahiplenme
Sürekli neden sorusuna yanıt aranır.
Yetersizlikler: Teknik bilgi ve tecrübe, yöntemler, eksiklikler, sistemler, arızalar, faktör ve etkenler.
Görüş alanını genişletmek.
Liderin tüm adımlar hakkında bilgi sahibi olması, farkındalığını artırması gerekir.

Süreçlerdeki sorumluların tümü dinlenmelidir. Problemin kaynağı, doğru soruyu bulmak, sorunun doğru kaynağını bulmak. Neden kaynak? Süreç doğru tanımlanmış mı? Sorun yaratan dış etmenler nelerdir? Kavramsal standartlara uyumluluk: test, boyutlar, biçimler, etkilenme.

Kök nedenler, problemin arkasında yatan gerçek sebeplerdir. Kök neden analizi, yaşanan problemlerin görünen nedenlerini ortadan kaldırmak yerine kalıcı bir şekilde çözüm üretmeye odaklanan bir süreç uygulamasıdır. Kök neden analizi, problemin çıkış noktasını tespit edebilmeyi amaçlar. Yine kök neden analizi, problemin öncelikli nedenini bulabilmek

için belirli aşamalarda ilgili araçları kullanır ve böylelikle; Ne olduğunu ve neden olduğunu saptamanızı sağlar.

Öngörülmemiş bir riskin doğmasından, faaliyetlerdeki başarısızlık, varlıkların zarar görmesi veya kaybı, güvenlik sorunları, kalitedeki düşüş veya hizmet sunumundaki memnuniyetsizliğe kadar birçok sorun alanında, kök neden analizi etkili bir şekilde yürütülebilir.

Temel nedenleri keşfetmeniz gerekir. Genellikle üç temel tür neden bulunur:

- Fiziksel Nedenler: Somut, materyal ögeler, bir biçimde başarısız olmuşlardır. (Örneğin, bir otomobilin fren sistemi bozulur).
- İnsani Nedenler: İnsanlar bir şeyi yanlış yapmıştır veya gerekli olan bir şeyi yapmamışlardır. İnsani nedenler genellikle fiziksel nedenlere yol açarlar (Örneğin, fren yağını kimse doldurmamıştır, bu da fren sisteminin bozulmasına sebep olmuştur).

- Organizasyonel Nedenler: İnsanların kararlarını verirken veya işlerini yaparken kullandıkları sistem, süreç veya politikalar hatalıdır. (Örneğin, araç bakımından sorumlu bir kişi belirlenmemiştir ve herkes fren yağını başkasının doldurduğunu düşünmüştür).Kök Neden Analizi, bu üç tip neden ile ilgilenir. Olumsuz etkilerin yapısının incelenmesini, sistemdeki gizli kusurların bulunmasını ve probleme katkıda bulunan belirli eylemlerin keşfedilmesini gerektirir. Bu, çoğu zaman şu anlama gelir: KNA birden fazla temel nedeni ortaya çıkarır.

Kök neden analizinin değerlendirilmesi.

- Problemi Anlama, Muhtemel nedenleri belirleme
- Bilgi Toplama
- Problem Veri Analizi

Kök neden analizi teknikleri.

- Balık kılçığı diyagramı,
- Beyin fırtınası,
- Ağaç diyagramı,
- Serpme diyagramı,
- Histogram,
- Kontrol grafikleri,
- Hata ağacı analizi,
- Hata türü ve etkileri analizi,
- Ishikawa diyagramı,
- Değişim analizi,
- Pareto analizi,
- 5 neden.

Kök Neden Analizi Süreci

Kök Neden Analizi'nin tanımlanabilir beş aşaması vardır.

1. Problemi Tanımlama: Neler görüyorsunuz? Belirgin semptomlar nelerdir?
2. Veri Toplama: Problemin var olduğuna dair hangi kanıtlarınız var? Problem ne zamandır var?
3. Probleme sebebiyet veren faktörler: Probleme sebebiyet veren faktörleri incelemeye başlamadan önce, durumun analizini tam olarak yapmalısınız. Kök Neden Analizi'nin etkinliğini maksimum düzeye çıkarmak için, durumu anlayabilecek herkesi bir araya getirin. Probleme aşina olan insanlar, sorunu kavrayabilmeniz için size yardımcı olabilirler.
4. Problemin muhtemel nedenlerinin belirlenmesi: Hangi olaylar dizisi probleme yol açmıştır? Problemin meydana gelmesine ne gibi durumlar sebep olmuştur? Temel problemin ortaya çıkışını çevreleyen diğer problemler nelerdir? Bu aşamada, mümkün olduğunca çok sayıda neden belirleyin. İnsanlar genellikle bir veya iki neden belirleyip bırakırlar, fakat bu yeterli değildir. Tanımlanan neden analizi ile yapılmak istenen şey, en belirgin nedenlere müdahale etmek değil; derinde yatan sorunları çözmektir.
5. Problemin kök nedenlerinin tanımlanması: Sebep olan faktör neden mevcut? Problemin ortaya çıkmasının asıl sebebi nedir?
6. Çözümin öne sürülmesi ve uygulanması: Problemin bir kez daha meydana gelmesini önlemek için ne yapabilirsiniz? Çözüm nasıl uygulanacak? Bunun sorumluluğu kime ait olacak? Çözümü uygulamanın barındırdığı riskler nelerdir? Neden-sonuç sürecini inceleyin ve çeşitli sistemler için gerekli olan değişimleri belirleyin. Çözümün etkilerini tahmin etmek için önceden plan yapmanızda fayda var. Böylelikle, muhtemel başarısızlıkları henüz ortaya çıkmadan saptayabilirsiniz.

Kök Neden Analizi, bir problemi anlamak ve çözmek için oldukça kullanışlıdır. Analitik bir araç olarak Kök Neden Analizi, önemli problemlerin kapsamlı ve geniş çapta denetlenmesinin yanında onlara yol açan olayların ve faktörlerin belirlenmesinde de kullanılan bir yöntemdir.

Balık Kılçığı Yöntemiyle Kök Neden Analizi Nasıl Yapılır?

1) İlk önce problem tanımlanır ve diyagramın sonuç kısmına yazılır.
2) Probleme etki edebilecek temel faktörler kılçıklara kutucuk şeklinde sıralanır.
3) Her bir faktörde sonuca etki edebilecek nedenler yazılır. Söz konusu nedenlerin asıl kaynağının tespiti için ard arda niçin sorusu sorularak her bir faktörde probleme yol açan temel kök neden tespit edilmeye çalışılır.
4) Tüm bu çalışma sonucunda, problem şeklinde ortaya çıkmış bir olay ile bu problemin nedenleri ayrıntılı ve kategorize edilmiş bir şekilde açıkça görünür hale gelir.
5) Ayrıntılı nedenler üzerinden önceliklendirme yapılarak hangi nedenler için ilk planda tedbir geliştirileceği kararlaştırılır.

Fabrika müdürü günlük olağan fabrika turunu atarken yerde birkaç yağ damlası görür. Temizlikçiyi çağırır herkesin içerisinde azarlar, hemen temizlemesini söyler ve gider. Temizlikçi temizler fakat yine yağ damlasını görür, korkar sürekli yağı temizlemeye devam eder. Sonra birgün sızıntı artar ve tavandaki borular patlayarak yangın çıkmasına neden olmuştur. Oysa müdür temizlikçiyi değil de ustabaşını çağırsa ve yağ damlamasının nedenini sorsa ve araştırılmasını isteseydi ustabaşı inceleme yaptığında hemen tepelerindeki kızgın yağ borularını birleştiren contaların sızıntı yaptığını söylerdi. Müdür contanın neden problem kaynağı olduğunu sorardı. Ustabaşı bakım kayıtlarına baktığında civatanın bir ayda dört defa değiştiğini görür ve rapor ederdi. "Neden dört defa değiştirildi?" Sorulduğunda bakımda çalışanların satın almaya şikayette bulunduklarını ancak, emir olduğunu belirtirlerdi. Satınalma sorumlusuna neden bu civataları almada ısrarcı oldukları sorulurdu. Fiyatın çok yüksek olduğunu, ürün imalatcısının fiyatı indirmediğini, bu nedenle finans müdürünün daha ucuz conta satın alınması emrini verdiğini beyan ederlerdi. Finans müdürüne neden ucuz civata almak için kaliteden feraget edildiğini sorduğunda finans müdürü: üç ay önce müdür olarak kendisinin maliyeti azaltmak için program oluşturduğunu söylerdi. Fabrika müdürü kök nedenin kendisi olduğunu fark ederdi.

4. Yazılım Geliştirmede Süreç Yönetimi

Yazılım geliştirme süreci, planlanmış bir iş akışıdır. Planlanmış bir iş akışı, zamanında ve kaliteli bir yazılımın elde edilmesinin sağlanmasıdır.

Süreç, işlevlerin belli bir taslağa uygun ve belli bir sonuca varacak biçimde düzenlenmesi, sıralanmasıdır. Bir şeyin yapılışını, üretiliş biçimini oluşturan sürekli işlemler ve eylemler dizisidir. Aralarında birlik olan veya belli bir düzen veya zaman içinde tekrarlanan, ilerleyen, gelişen olay ve hareketler dizisidir.

Yazılım Süreci: Bir yazılım ürününü üretmeyi sağlayan birbiriyle tutarlı aktivite grubudur.
Ne yapılmak istendiğini tüm uygulama detaylarına girmeden tanımlar. Yazılım süreci, yazılım mühendisliğinin yazılım üretmeye yönelik yol haritasıdır.

Bilginin, kültürün, hizmetin veya teknolojinin üretilmesine katma değer denir. Şu ana kadar arazi ve makine önemli sermaye birikimi olarak karşımıza çıkmaktadır, bundan sonra bilgi çok önemli bir sermaye birikimi olacaktır. Yazılım, özelllikle kod yazılım stratejik bilgi birikimini gerektirmektedir. Yazılım, istenen nicelikte, istenen zamanda ve istenilen nitelikte yapılmak zorundadır. Bu zorunluluğu üzerine alan kişi ya da gruplardan oluşan paydaşların amacı süreçi planlamalı, yapılışını ve niteliklerini denetlemeli, yöntemlerini irdelemeli, işletme düzenini ve malzeme akışını kontrol etmelidir.

Çeşitli modellerin kendine özgü avantaj ve dezavantajları vardır. Gerçeklenecek projeye uygun modelin seçilmesi gerekir.

Yazılım Geliştirmede Süreç yönetimi:

- Süreç akış yöntemleri, süreçler arası ilişkilerin ve iletişimin gösterildiği yöntemler (Veri Akış Şemaları, Yapısal Şemalar, Nesne/Sınıf Şemaları).
- Süreç tanımlama yöntemleri, Süreçlerin iç işleyişini göstermek için kullanılan yöntemler (Düz Metin, Algoritma, Karar Tabloları, Karar Ağaçları, Anlatım Dili).
- Veri tanımlama yöntemleri, Süreçler tarafından kullanılan verilerin tanımlanması için kullanılan yöntemler (Nesne İlişki Modeli, Veri Tabanı Tabloları, Veri Sözlüğü).

Yazılım Geliştirme Süreç Yönetimi,

- Endüstri kaliteye önem vermektedir. (Performans, Üretkenlik)
- Deneyimler göstermektedir ki, Yazılım Geliştirme süreçlerinin yazılımın kalitesine kayda değer etkisi vardır. **Yazılımın istenen kalitede olmasını süreçleri kontrol ederek ve süreçleri iyileştirerek sağlayabiliriz.**
- Yönetici ve geliştiricilerin, *yazılım geliştirme sürecinin karışıklığı ile baş etmelerini* sağlar.

İyi yapılandırılmış yazılım mühendisliği ilkelerini desteklemek için tasarlanmış operasyonel bir süreç olan TSP'yi (Team Software Process for Secure Software Development) kullanmak için, yazılımcıların önce Personal SoftwareProcess (PSP) denilen kişisel yazılım süreci konusunu bilmeleri gerekir. Bunlar entegrasyon testi, sistem testi, saha testi ve müşteri kullanımında hata tespit etme olarak sıralanır. Amaç kod yazılım başına hata sayısını azaltmaktır.

Her hatayı bulup çözmenin maliyeti yaklaşık olarak bir günlük mühendislik çalışma saattir. Birim testi yapmanın maliyeti ise yaklaşık olarak mühendislik çalışma saatinin 4 ile 5 katıdır. Hatayı düzeltmenin kodlama maliyetinden 5 kat daha fazla olduğu sonucu ortaya çıkar.

Uygulamaya özgü test geliştirilmeli, dokümantasyon, tekrarlı geliştirme, kod yeniden yapılandırma ve test anlayışları noktalarında çakışmalar ortadan kaldırılmaldır. Bunlar;

- Doğrudan iletişim ve üstü kapalı bilgi,
- Kısa ve sık iterasyonlar,
- Paylaşımlı kod sahipliği ve aktif kod yeniden düzenleme (refactoring) ile tasarımın aşamalı ortaya çıkışı,
- Kullanıcı hikâyeleriyle karar verilen, test durumları ile oluşturulan test tabanlı geliştirme.

Öncelikle projenin sonlarına doğru ortaya çıkabilecek güvenlik açıklarının olma olasılığını azaltmaktır. Bu amaçla,

- Geliştirme takımı içerisinde, riskler belirlenir, kullanıcı hikâyelerinin önemsenir, kodlama ile sistemin tasarım ve kodlama aşamasında gerçek zamanlı risklerin gözden geçirmelerini yapması için bir test mühendisi olmalıdır.
- Risk argümanı oluşturmak için, test mühendisi programlama aktivitelerini dokümente etmelidir.
- Risk argümanı oluşturmak için, test mimarisi dokümente edilmelidir.
- Programlama statik doğrulama ve otomatik yürütmeler ile tamamlanmalıdır.

Risk gereksinimleri belirleme sürecini kötüye kullanım senaryoları ve test ile alakalı kullanım senaryolarının oluşturulmasıda dâhil edilmelidir. Senaryolardaki tehditler ve buna karşı alınacak önlemlere karşılık olarak belirlenen gereklikodlama ve tasarım standartlarının ilgili senaryolarla eşleşme yapmaya yardımcı olacaktır.

Paydaşlar tarafından oluşturulan gereksinimlerden riskler için hikaye oluşturup eğer kritikse devam etmekte olan kod yazılımının sonlandırılarak yenidenplanlaması risk gereksinimlerinin gerçekleştirimi ve testininyapılmasını, kritik değilse sonraki kod düzenlenmesine kadar beklemesi gerekir. Risk tanımlama işlemi, paydaşlar tarafındanöneme sahip parçalara odaklanarak, riskle alakalı kullanıcı hikâyeleri tanımlama

üzerine kuruludur. Örneğin, arabellek taşması, gizlilik(confidentiality), bütünlük (integrity), kullanılabilirlik (availability) gibi güvenlik riski oluşturan özellikler bağımsız birer güvenlik problemidir. Göz önünde bulundurulması gereken ise, yazılım güvenliğinin, riskedayalı güvenlik gereksinimlerinin belirlenip önleminin alınmasıyla beraber, yazılımiçerisinde güvenlik sorunlarına yol açacak, programlama veya tasarım hatalarındanoluşabilecek içsel tehditlerin (insider) de irdelenmesidir. Sadece paydaşlarla belirlenen kullanıcı senaryolarında güvenlik olaylarına odaklanmakbunu sağlayamaz. Tüm yazılımı ilgilendirenve geliştirme süreci boyunca hesaba katılan bir nitelik olarak bakmaktadır ve buna yönelik uygulamalar içermektedir.

İçsel tehditler kategorisinde yer alan kodlama hataları gibi istenmeden yapılan veya eğitim eksikliği nedeniyle oluşan gerçekleştirim ve sistem hataları vb. hatalarda hesaba katılmalıdır. Risk güvenlik uygulamalarından olan eğitim, kodlamastandartları, güvenli tasarım ilkeleri, kod ve tasarım gözden geçirmeler, statik analizaraçlarının kullanımı gibi uygulamalarda göz önünde bulundurulmalıdır.

Güvenlik test yazılım geliştirme süreç modellerinde bulunması gerekenaşamalar aşağıda verilmiştir:

- Güvenlik ekibi yazılım geliştirme organizasyonu bünyesinde oluşturulmalı bu yüzden tüm proje ekibi üyelerine güvenlik eğitimi verilmelidir. Gerektiğinde sürecegüvenlik uzmanı dâhil edilmelidir.
- Güvenlik gereksinimleri belirlenmeli ve kötüye kullanım senaryolarıoluşturulmalıdır.
- Tasarım aşamasında gerçekleştirilmesi gereken güvenlik eylemlerine değinilerek eniyi tasarıma karar verilmelidir.
- Risk analizi yapılmalıdır.
- Kodlama standartlarına ve güvenli kodlama ilkelerine uyulmalı, birim testleriyazılmalıdır.
- Statik analiz araçları kullanılmalıdır.
- Kod ve tasarım gözden geçirme yapılmalıdır.
- Güvenlik testi, nüfuz testi, bulandırma testi yapılmalıdır.
- Kod yeniden yapılandırma ile güvenlik eklemeleri, düzeltmeleri yapılmalıdır.
- Güvenlik dokümanları oluşturulmalıdır. Bu dokümanlar kullanıcının hata yapmaması için kullanıcı kılavuzu gibi güvenlik sorunlarını önlemede ve geliştirici için tasarım dokümanları, yorumlarla desteklenmiş kaynak kod ve test senaryoları gibi güvenlik geliştirmesinde etkili olabilecek dokümanlardır.

4.1. Yazılım Yaşam Döngüsü

Yazılım yaşam döngüsü, herhangi bir yazılımın, üretim aşaması ve kullanım aşaması birlikte olmak üzere geçirdiği tüm aşamaları tanımlar.

Yazılım işlevleri ile ilgili gereksinimler sürekli olarak değiştiği ve genişlediği için, söz konusu aşamalar bir döngü biçiminde ele alınır.

Döngü içerisinde herhangi bir aşama da geriye dönmek ve tekrar ilerlemek söz konusudur.

Yazılım yaşam döngüsü tek yönlü ve doğrusal olduğu düşünülmemelidir.

Yazılım Yaşam Döngüsü Temel Adımları:

1. **Planlama**
2. **Çözümleme**
3. **Tasarım**
4. **Gerçekleştirme**
5. **Bakım**

Planlama: Üretilecek yazılım ile ilgili olarak, çalışma ortamı, personel ve donanım gereksinimleri ile ilgili olarak fizibilite çalışmasının yapılarak detaylı planının oluşturulduğu aşamadır.

Çözümleme: Yazılım işlevleri ile gereksinimlerin ayrıntılı olarak çıkarıldığı aşamadır. Bu aşamada temel olarak mevcut sistemde var olan işler incelenir, temel sorunlar ortaya çıkarılarak, yazılımın çözümleyebilecekleri vurgulanır. Temel amaç, bir yazılım mühendisi gözüyle mevcut yapıdaki işlerin ortaya çıkarılması ve doğru olarak algılanıp algılanmadığının belirlenmesidir. Bu aşamada temel UML diyagramlarının çizimine başlanır (Use Case, Activity, Class diagram… vs.)

Tasarım:

- Çözümleme aşamasından sonra belirlenen gereksinimlere karşılık verecek yazılım ya da bilgi sisteminin temel yapısının oluşturulması çalışmalarıdır.
- Bu çalışmalar, mantıksal tasarım ve fiziksel tasarım olarak iki gruba ayrılır.
  - **Mantıksal Tasarım:** Mevcut sistem değil önerilen sistemin yapısı anlatılır. Olası örgütsel değişiklikler önerilir.
  - **Fiziksel Tasarım:** Yazılımı içeren bileşenler ve bunların ayrıntıları içerilir.

Gerçekleme: Kodlama, Test etme, Kurulum çalışmalarının yapıldığı aşamadır.

Bakım:

- İşletime alınan yazılım ile ilgili olarak, hata giderme ve yeni eklentiler yapma aşamasıdır.
- Bu aşama yazılımın tüm yaşamı boyunca sürer.

4.2. Risk Analizi

Risk analizi sistemdeki riskleri değerlendirip analiz etmeye yönelik bir aktivitedir. Riskanalizi yapılırken yazılımın spesifikasyonları, tasarımı, mimari dokümanları, verihassasiyeti gibi konuların üzerine yeterli araştırma yapılıp bilgi sahibi olunmalıdır.

Yazılımın karşılaşabileceği güvenlik olayları ve yazılımdaki kusurlar (flaw) belirlenir. Organizasyon değerlerini tehlikeye düşürebilecek potansiyel tehditler ve bunlara ilişkin riskler analiz edilir. Tehlikenin gerçekleşmesi durumunda ne tür etkilere neden olacağına dair analiz yapılır. Riskler etkilerine göre önceliklendirilir. Riski azaltmak için alınan önlemleri içeren bir hafifletme stratejisi geliştirilir. Yazılım, bu riski hafifletmeçalışmalarının etkisini tespit etmek için yeniden değerlendirilir. Tüm araştırmalar, bulgularve değerlendirmeler rapor edilir.

Risk analizi yapılırken özet olarak organizasyon değerleri, bu değerlere karşı olabilecek tehditler, sistem açıkları, riskler ve olasılıkları belirlenir. Risk analizi planlı, olaya dayalı veya ihtiyaca dayalı esaslarla yapılabilir.

Risk analizindeki bazı kavramlar aşağıda verilmiştir:
Değer: Organizasyonun korunan değerleri, veri bileşeni veya tüm sistem.
Risk: Bir değerin olumsuz bir etkiden değer kaybına uğrayacağının olasılığı. Birçok faktör bunu belirleyebilir: uygulamanın kolaylığı, saldırganın motivasyonu ve kaynakları, sistemin varolan açıkları.
Tehdit: Zararlı etkenin neden olacağı tehlike.
Sistem açığı: Hata, zayıflık.
Etki: Riskin gerçekleşmesi durumunda organizasyona etkisi. Parasal veya itibarla ilgili olabilir veya kanun, düzenleme, sözleşme ihlalleri ile sonuçlanabilir.
Olasılık: Verilen olayın gerçekleşme olabilirliği.

Tehditler sistemin ve değerlerin korunmasını ve güvenlik ilkelerini ihlal eden tehlikedir. Tehditler kod kırıcıları ve casuslar tarafından kaynaklanan bilinçli tehditler olabileceği gibi veri giriş hataları, programlama hataları veya eğitim yetersizliğinden kaynaklanan istenmeden yapılan hatalar da olabilir.
İstenmeden yapılan programlama hataları da bu içsel tehditlere dâhildir.
Risk yönetimi yazılım yaşam döngüsü boyunca riskleri yeniden değerlendiren sürekli bir işlemdir.

- Proje başlangıç aşamasında, değerler tanımlanır ve bu değerlerin zarar görmesi sonucu etkileri belirlenir. Riskler sistem gereksinimleri ve güvenlik operasyon konsepti bakımından değerlendirilir.
- Geliştirme aşamasında, belirlenen riskler yazılımın güvenlik analizine desteksağlar.
- Gerçekleştirme aşamasında, risk yönetimi, sistemin operasyonel ortamdagereksinimlere göre gerçekleştiriminin değerlendirilmesine destek sağlar.
- Bakım ve operasyon aşamasında, periyodik olarak yapılan sistemin yenidenyetkilendirilmesi (veya onaylanması) veya yazılımda büyük değişiklikler yapılmasıdurumlarında yeniden güvenliği değerlendirip sağlamak adına risk yönetimiaktiviteleri yapılır.

Değerler, tehditler ve olası sistem açıkları belirlendikten sonra yazılımın atağa açık alanlarını en aza indirgeyecek şekilde tasarım yapılır. Tahdit alanları, kod parçası, arayüz, servis, protokol (iletişim kuralı) ve özellikle yetkisiz kullanıcılarolmak üzere tüm kullanıcılara açık olan uygulamalar olarak tanımlanmıştır. Bu alanıminimize etmek de yazılımda oluşabilecek olası atakları azaltmak anlamına gelir.

4.3. Güvenli Kodlama ve Test

Güvenli kodlama ilkelerine göre geliştirme yapılmalı, statik analiz araçları ile kod taranmalı, kod gözden geçirmelerle araçların bulamadığı hatalar bulunmalıdır. Teste oldukça çok yer verilmeli, hata bulundukça bunun için test yapılmalıdır. Bunlar güvenlik için de geçerlidir. Örneğin tam sayı taşması olursa bunu tetikleyen güvenlik testi yazılmalıdır. Test, kodun her derlenmesinde çalıştırılarak hatanın giderildiğinden emin olunmalıdır. Tüm hatalar ve güvenlik hataları için testler her çalıştırıldığında başarılı olmalıdır.

Sistem açıkları gerçekleştirim seviyesindeki ve tasarım seviyesindeki hatalar olmak üzere iki kategoriye ayrılabilir. Gerçekleştirim seviyesindeki hatalar için kaynak kod analizi yapılır. Tasarım seviyesindekiler için ise güvenli tasarım ilkeleriyle ilgili kaynaklardan ve organizasyonun rapor ettiği tecrübelerden yararlanılır. Yazılımlarda ciddi güvenlik açıklarına neden olan ve oldukça yaygın olan gerçekleştirim hataları aşağıda verilmiştir:

Yarış Durumu (Race Condition): Doğru senkronize edilmemiş işler (thread) arasındaki zamanlama problemlerinden ortaya çıkan, aynı zaman diliminde bir kaynağa erişme mücadelesi olarak tanımlanabilir. Herhangi bir zamanaşımı kontrolü olmadan yazılımının hiç bırakılmayacak bir kaynağı beklemesi (deadlock), paylaşılan kaynaklara eşzamanlı

erişim hataları (resource collision), programın sonlanmasına izin vermeyen mantıksal veya kontrol akışlar, sonsuz döngüler (infinite loops) örnek verilebilir.

Girdi Geçerleme (Input Validation): Kullanıcı veya parametre girdilerine güvenmek sorunlara yol açabilir. Semantik veya SQL enjeksiyonlar gizlilik (confidentially) ve bütünlük (integrity) için risk oluşturur. SQL enjeksiyon veritabanındaki hassas bilgiler için tehlike oluşturur.
Olağandışı Durumlar (Exceptions): Kodun normal akışını bozan olaylardır. Exception handling programın anormal olayları atlatmasını sağlar.

Bellek Taşması (Buffer Overflow): C, C++ kodlarının, dizin (array) ve işaretçi (pointer) yapılarında sınır kontrolleri olmadığı için emniyetsiz oluşundan dolayı geliştirici bu kontrolleri yapmalıdır. Bu açıktan yararlanan saldırganlar uzaktan zararlı kod enjeksiyonu ile yazılımı kötüye kullanabilir.

Yığın Taşması (Stack Overflow): Bellek taşmasının bir formu olan yığın taşması, programın çalışma zamanı için ayrılan belleğin dikkatsiz kullanımıdır. Saldırgan kontrolün zararlı koda geçmesini sağlayabilir.

Tamsayı Taşması (Integer Overflow): Tamsayıyı ikili gösterime yetmeyecek bir alana yerleştirme girişiminde ortaya çıkar.

Statik analiz araçları kaynak kodu tarar ve derleyicinin yakalayamadığı daha sonra problem yaratabilecek hataları bulur. Elle yapılan denetimler tek başına çok zaman alıcı olabileceği için statik analiz araçları hızlı bir şekilde kodu tarayarak belirli hataları bulmada oldukça etkilidir.

Kod gözden geçirme, mantıksal hataları ve statik analiz araçlarının bulamadığı hataları bulmada etkilidir. Risk analizi ile beraber kaynak kod gözden geçirmesi en iyi yazılım güvenlik pratikleri arasında yüksek öneme sahiptir [64]. Usulüne uygun gerçekleştirilen tasarım ve kod gözden geçirme, yazılım güvenliğinde önemli ölçüde iyileştirmeler sağlar. Gözden geçiriciler kodlama standartlarını dikkate alarak kodu inceler, birim test planı yapar, gereksinimlere uygunluğu kontrol eder, işlevselliklerin doğru gerçekleştirildiğini kontrol ederler. Yorum, belgeleme, birim test planı, gereksinimlere uygunluk gibi özelliklerin listelendiği kod gözden geçirme kontrol listelerini kullanırlar.

Güvenli kodlama ve test ilkeleri aşağıda maddeler halinde verilmiştir:

1) Kodlama ve test standartlarının uygulanması: Kodlama standartları, yazılımcının güvenlik açıklarına neden olacak hatalar yapmasını önler. Örneğin, güvenli bir dizgi ele alma (string handling) ve arabellek işletmesi (buffer manipulation), arabellek taşması (buffer overrun) zafiyetlerinin meydana gelmesini önler.
2) Test standartları sadece yazılımın doğru işleyişine yoğunlaşmaktan ziyade, testin potansiyel güvenlik açıklarını bulmasına da odaklanır.

3) Bulandırma araçlarını (fuzzing tools) içeren güvenlik testi araçlarının kullanılması: Fuzzing yazılıma geçersiz girdiler verip test edilmesini sağlayarak, zafiyete yol açacak hataların bulunmasını azami seviyeye çıkarır.
4) Statik kod analizi yapan araçların kullanılması: Bu araçlar, arabellek taşmaları (buffer overruns), tamsayı taşmaları (integer overruns) ve ilk değer verilmemiş değişkenler (uninitialized variables) gibi zafiyetlere neden olabilecek kodlama hatalarını tespit eder.
5) Kod gözden geçirme uygulamalarını geliştirilmesi: Kod gözden geçirme, yazılımcıların kaynak kodu incelemesi ve potansiyel güvenlik açıklıklarını bulup yok etmesini sağlayarak otomatik araçlara ve testlere destek olur. Mantıksal hataları ve otomatik araçlarının bulamadığı hataları bulmada etkilidir.

Yazılım Testi

Güvenlik testi, hem güvenlik özellikleri fonksiyonlarının gerçekleştirildiğinin hem de saldırganın bakış açısıyla sisteme bakıp olası sistem açıklarının hesaba katılarak yapıldığının testini gerektirir. Güvenliğin sadece güvenlik mekanizması ve özelliklerinden ibaret olduğu konusundaki yanlış kanı, güvenlik testinde de yaygındır. Güvenlik, tüm sistemi ilgilendiren gerekli bir niteliktir. Örneğin bellek taşması, güvenlik özelliklerinde veya kritik bir arayüzde ortaya çıkmasından bağımsız olan bir güvenlik problemidir. Bu yüzden güvenlik testi aşağıdaki iki yaklaşımı içermelidir:

- Güvenlik mekanizması fonksiyonlarının doğru gerçekleştirildiğinin testi,
- Saldırganın bakış açısıyla bakıp sistem açıklarını göz önünde bulundurarak yapılan riske dayalı güvenlik testi.

Güvenlik mekanizması fonksiyonelliği, geleneksel yaklaşım kullanan standart test organizasyonları tarafından yapılabilir. Örneğin erişim denetim mekanizmasının beklendiği gibi çalıştığının testi standart bir işlemdir. Geleneksel kalite güvence ekibi risk tabanlı güvenlik testi için zorlanabilir. Çünkü bunun için tasarımcı saldırgan gibi düşünebilmelidir. Ayrıca güvenlik testleri direkt olarak güvenlik istismarı yaratmayabilir ve gözlemlenmesi gerekebilir. Güvenlik testi, testçinin daha sonra kapsamlı, detaylı analiz yapmasını gerektirecek öngörülemeyen sonuçlar verebilir. Kısacası risk tabanlı güvenlik testi uzmanlık ve deneyim gerektirir.

Minimum insan müdahalesi gerektiren ve risk analizlerine dayanan bir yapıda ilk test seti fonksiyonel güvenlik test setidir ve sınıf kodları ve yorumları kripto fonksiyonelliğini test etmek üstüne kurulmuştur. Çoğu kartlar bu fonksiyonel güvenlik testlerinin hepsini geçmiştir. Ancak risk tabanlı test yaklaşımına göre saldırgan uygulama takımı ile yapılan testlerde tüm kartlar başarısızlık sergilemiştir.

4.4. Kod Yeniden Yapılandırma (Refactoring)

Kod yeniden yapılandırma, kodu daha temiz okuması ve devam ettirmesi daha kolay, daha kaliteli ve hatta daha güvenli yapmak için iç gösterimini iyileştirmektir. Güvenlik gereksinimi olarak, sürekli olarak sistematik bir şekilde eski kodlar gözden geçirilir, güvenlik hataları aranır, eğer bulunursa düzeltilir. Kod yeniden yapılandırma yapılırken kodun arayüzü veya davranışları değiştirilmeden, kodun kendisinde yeniden yapılandırma ve değişimler yapılır. Güvenlik hataları, kod yeniden yapılandırma işleminin bir parçası olarak görülmelidir.

Eğer mevcut kod hassas ve kişisel veri işliyor ve bu veri internete açılıyorsa, tüm kod gözden geçirilmeli, tüm kod yeniden analiz edilip yeni testler oluşturulmalı ve tüm hatalar giderilinceye kadar yeniden yapılandırma işlemleri yapılmalıdır. Sahaya sürülmeden önce kodun güvenlik açısından gözden geçirilip yeniden yapılandırılması güvenliği sağlama konusunda oldukça fayda sağlar.

Güvenlik hataları;

- Güvenlik alanı, hem olumlu yönden hem de olumsuz yönden sürekli olarak önemli derecede gelişmektedir, daha çok da olumsuz yönden ilerlemektedir.
- Güvenlik araçları, olumlu ve olumsuz yönden çok çabuk gelişmektedir. İnsanlar, güvenlik hatalarını bulmada daha iyiye gitmektedir.

4.5. Yazılım Geliştirmenin Stratejik Aadımları

Yazılım geliştirmenin ilk adımı stratejik adımları ve yönergelerin dokümante edilmesidir.

Mühendislik

Paydaşların istekleri ile süreç ve işlevlerin gereksinimleri belirlenir. Ürün ve ürün bileşenleri oluşturulur. Ürün bileşenlerinin yazılım mimarisi içerisindeki yerleri ve arayüzleri belirlenir, analiz edilir ve raporlanır. Operasyonel içerik ve senaryolar oluşturularak işlevler ortaya çıkartılır. Maliyet, zaman, performans, işlevler, bileşenler, hassasiyetler ve riskler öngörülerek plan oluşturulur.

Teknik çözümü oluşturacak araştırma ve tasarım uygulamaları geliştirilir. Alternatif çözümler araştırılır ve değerlendirilir. Belirlenmiş çözüm önerileri detaylı tasarlanır. Oluşturulan tasarım, ürün ve ürün bileşenlerinden oluşur.

Bu süreç sonunda, üst düzey yazılım tasarımı, detaylı yazılım tasarımı, sistem tasarımı, kullanıcı arayüz tasarımı, veritabanı tasarımı, veri aktarma tasarımı, test planı ve test senaryoları üretilerek dokümante edilir.

Yazılım ürünlerinin, paydaşların isteklerinin tamamını karşıladığını ve amacına uygun olduğunu analiz edilerek ve onaylanmasıdır. Bu süreçte, paydaşların önerileri

değerlendirilerek düzeltici faaliyetler planlanır. Tüm dokümanlar özellikle Konfigürasyonlar kurumsal hafıza yönetimince saklanır.

Yazılımın doğruluğu, tutarlılığı, izlenebilirliği, süreç ve işlevlerin uygunluğu gözden geçirilerek kalite güvencesi sağlanır. İş geliştirme süreci boyunca proje ile ilintili tüm planlar, gereksinim kayıtları, taasarımlar, test senaryoları gibi yüm unsurların doğrulama kayıtları üretilir.

Projenin ürünleri ve ürün bileşenlerinin gereksinimleri ile iş ürünleri ve proje planı arasındaki tutarsızlıklar araştırılır. Bu araştırma sürecinde:

- Gereksinimlerin önemi hakkında fikir oluşturulur.
- Paydaşlarda farkındalık yaratılır, gereksinimlerin onayı sağlanır.
- Gereksinimlere ait tüm değişiklikler izlenir ve yönetilir.
- Gereksinimlerin, proje planı ve iş ürünleri ile tuturalılığı sürekli takip edilir. Tutarsızlıklar belirlenir. Düzeltici etkilikler planlanır ve yürütülür

Proje Yönetimi

Proje yönetimi için öncelikle izleme ve kontrol süreci oluşturulur. Proje yönetiminin amacı, plana göre süreç ve işlevlerin koordinasyonu ve yönetimidir. Proje yönetiminde kaynaklar tanımlanır, proje aktiviteleri planlanır, proje maliyet hesapları yapılır, hedefler ve sapmalar sürekli izlenir. Bir sapma durumunda düzeltici etkinlikler devreye alınır. Proje yönetim aşamalarında proje planı, raporları, değişiklik isterleri gibi yönetimsel dokümanlar üretilerek proje paydaşları ile paylaşılır.

Risk yönetiminde ürünün veya projenin hedeflerine ulaşmasındaki olumsuz etkilerin iş üretme sürecini ve işlevlerini etkilememesi için, gerekli olan önlemlerin belirlenip planlanmasıdır. Potansiyel problemler oluşmadan önce tanımlanır. Risklerin tanımlanması, önceliklendirilmesi, risk azaltma tekniklerinin araştırılması, beklenmedik durum planlarının oluşturulması, risklerin ölçüm parametrelerinin tanımlanması, beklenmedik durum planlarının uygulanmasını kapsar. Risk yönetimi idari ve teknik işlevleri, sürekli ileriye baktıran bir süreçtir.. Risk kataloğu ve risk planı bu sürecin çıktı iş ürünleridir.

Tedarik edilecek ürün ve hizmetleri seçim kriterlerinin oluşturulmalıdır. Tedarik edilecek değerlerin kaynağından itibaren ve kabul sürecine kadar yönetilmesi gerekmektedir. Sözleşme ile imza altına alınan paydaş isteklerinin, Alt Yüklenici tarafından sağlanmasının takibi, değerlendirilmesi ve kabul edilmesidir. Bu süreç yöetimi ile Alt Yüklenicinin üretim ve geliştirme süreci kontrol altında tutulur. Alt Yüklenici sözleşmeleri, değerlendirme kayıtları, konfigürasyon birimleri değerlendirme kayıtları gibi dokümanlar bu sürecin çıktılarıdır.

Destek süreci, konfigürasyon yönetimi, karar analizi ve çözüm süreci, ölçümleme ve analiz, kalite güvence süreci gibi alt süreçlerden oluşur.

Konfigürasyon yönetimi amacı, iş süreçleri döngüsü içerisinde üretilen tüm yazılımlarda yapılacak değişikliklerin kontrol ve yönetimini sağlamaktır. Bu süreci uygulanması ile, bir proje esnasında üretilecek tüm yazılım, donanım, altyapı, lisanslar, sözleşmeler ve iş ürünleri belirlenir ve tanımlanır. Süreçlerin ve işlevlerin durumları ve değişiklik istekleri kaydedilir ve raporlanır, saklanır. Konfigürasyon planı, konfigürasyon birimleri, değişiklik istekleri, etki analiz raporu bu sürecin çalışması ile üretilebilecek dokümanlardır.

Karar analizi ve çözüm sürecinde, konulmuş olan kriterlere göre değerlendirme yöntemleri oluşturmak, analiz etmek ve karar vermektir. Bu süreç ile alternatif çözümler belirlenir ve dokümante edilir.

Ölçümleme ve analiz sürecinde proje ilerleyişiyle ilgili gerçekleşen veriler toplanır, sınıflandırılır, raporlanır. Grafiksel izleme arayüzleri oluşturulur.

Kalite güvencesini sağlama sürecinde yürütülen tüm etkinliklerin belirlenen kalite sistemine uyumunu denetlenerek raporlanır. Proje planının yazılım geliştirme standartlarına uygun hazırlanması, projenin ilerlemesi esnasında tanımlı yazılım ürünlerinin üretilmesi, çıktıların kalite sistemi standartları ve kriterlerine uyumu sağlanmış olur.

Süreç Yönetimi

Yazılım geliştirme süreçleri sürekli yaşayan ve iyileştirilen süreçler olmak zorundadır. Kurumsallığın amacı, kurumun iş süreçlerinin tutarlı, tekrarlanabilir, performansını destekleyecek kurum hafızası oluşturmaktır.

Organizasyonel eğitimin amacı; çalışanların , kendilerine verilen rolleri verimli ve etkin bir şekilde gerçekleştirmeleri için; bilgi ve beceri düzeylerinin geliştirilmesinin sağlanmasıdır.

Kurumsal iyileştirme sürecinin amacı; kuvvetli ve zayıf alanlar baz alınarak; süreç ve işlevlerde iyileştirme aktivitelerinin planlanması ve uygulanmasıdır.

Yazılım süreçlerinin genel adımları

Çözümleme (Analysis)
Tasarım (Design)
Gerçekleme (Implementation)
Sınama (Testing)
Bakım (Maintenance)

ÇÖZÜMLEME

Çözümleme: Bir şeyi anlayabilmekiçin parçalarına ayırmaktır.Gerçeklenecek sistemi anlamayayönelik çalışmalardan ve **üst düzey** planlama eylemlerindenoluşur.

- Uygulama alanı
- Kullanıcı gereksinimleri
- Program parçaları arasındaki üst düzey ilişkiler ve etkileşimler (NYP'deki parçalar: sınıflar ve nesneler)

“Bir sorunu anlamadan, ölçeklendirmeden çözemezsiniz.”

TASARIM

Tasarım: Bir araştırma ve/veya geliştirme sürecinin çeşitli dönemlerinde izlenecek yol ve işlemleri tasarlayan çerçeve.Çözümleme ile anlaşılan sorun tasarım aşamasında kağıt üzerinde çözülür.

Yazılım, Tasarıma yönelik şemalar (NYP'de bazı tür UML şemaları) ile etkileşimlidir, elektronik, devre şemaları ile, mimari ise kat planları ile etkileşimlidir.

GERÇEKLEME

Eldeki tasarım, bir programlama dili ile kodlanır.

SINAMA

Sınama önemlidir, çünkü yazılım sürecinde ilerledikçe, ortaya çıkabilecek hataların giderilme maliyeti üstel olarak artar.Aksi gibi, hataların büyük çoğunluğunun kökenleri isteklerin belirlenmesi ve tasarım aşamalarındaki sorunlara dayanır.Bu yüzden, erkenden, sık sık ve kolay sınama yapılmalıdır.

BAKIM

Yazılımın faaliyete geçirilmesinden sonra sistemde yapılan değişikliklerdir.
Yazılım hatalarının düzeltilmesi:

- Kodlama hataları
- Tasarım hataları (!)
- Gereksinim ve analiz hataları (!!)

Bu aşamada sistemin işlevleri değiştirilir veya işlevlere eklemeler/çıkarmalar yapılır. Yazılım farklı bir ortama taşınır (programlama dili, işletim sistemi, donanım, iklim, vb.) (porting).
Yeniden mühendislik (Refactoring / Software re-engineering) yapılır. Tersinden mühendislik anlayışı geliştirilir. Yazılımın işlevini değiştirmeden iç yapısını değiştirmek için teknik bakış açısı geliştirilir.

Olası işler, yazılımın belgelendirilmesi, tasarımın iyileştirilmesi/değiştirilmesi ve yazılımın farklı bir ortama taşınması olarak sıralanır.

Yazılım sürecini oluşturan bileşenler

Proje Yönetimi ve Yönetim Rolleri
Proje Görünürlüğü, Karmaşıklığı, Büyüklüğü
Altyüklenici ya da dış işçilik isteği
Müşteri Katılımı
Organizasyon: Girişimci, Makine, Profesyoneller, Farklılaşmış, Yenilikçi
Proje Elemanlarının Rolleri, Tecrübe ve Yetenekleri
Ekip Çalışması, Büyüklüğü, Elemanların Alan Bilgisi
İş birliği
Proje Elemanlarının Yerleşimi

Yazılım Nitelikleri, Kalite, Güvence ve Hata Toleransı
Değer Üretimi
Kullanıcı Deneyimi
Dokümantasyon ve Modelleme
Teslimatlar
Yeniden Kullanılabilirlik
Kullanım Onayı
İhtiyaca Yabancılaşma
Değişime Açıklık
Gereksinimlerin Anlaşılırlığı, Uygulanabilirliği ve Güvenilirliği
Kritik Yönetimler: Gereksinimler, Zaman, İş Gücü ve Risk

4.6. Yazılım İyileştirme Süreci

Yönetimin yazılımı sahiplenmesi ve kaynak sağlaması iyileştirme sürecinin başarılı olabilmesi açısından kritik öneme sahiptir. Yazılım süreci başladığı anda, otomatikmen değişimler de başlamış olur. Değişime hazırlanmada kazanımları, maliyetleri, tehditleri ve fırsatları anlatan raporlar çok faydalı olacaktır.

Süreçleri ve işlevlerini tanımlamada iyileştirme ekibi "Yazılım Süreç Tanımlama" veya "Süreç İyileştirme Yönetimi" gibi konularda yetenek sahibi olmak zorundadır. Gerekli eğitimler önemlidir.

Yazılım sürecinde işlevler yerine getirilirken durum tespitinde fırsatları ve engelleri tanımlayabilmek adına veri toplama ve paydaşlar arasında sağlıklı bir iletişim ortamı oluşturulmalıdır. Karışıklık fark edildiğinde, kriz anında, hedef giden yolda sapma olduğu görüldüğünde, kimin ne ne yaptığı belli olmadığında ya da bulunulan noktada ne durumda olunduğu görülmek istendiğinde anlık durum tespiti yapılır. Anlık durum tespitinde tüm paydaşlar bilgi sahibi olmalıdır. Süreçlere, işlevlerine ve çalışanlarına müdahaleci yaklaşımın engellenmesi gerekmektedir.

Hedefe ilerlerken nerede olunmak istendiği; bunun karşılığında nerede olunduğu belirlenmek istenir. Bu aşamada her bir seviyenin hedefleri farklı olabileceğinden yöneticiler, proje liderleri ve çalışanlar için hedef belirlemede bir denge sağlamak çok önemlidir. Bu belirlendikten sonra süreçler arasında öncelik belirlenir ve iyileştirme planı hazırlanır. Yapılan çalışmalar hazırlanan plan çerçevesinde izlenir.

Tüm süreç boyunca dürüst ve iletişime açık bir ortam oluşturulmalıdır. Muhbirlik olarak adlandırılsa da geri bildirim, önemsenmelidir.Geri bildirim, paydaşlardan, denetcilerden, gözlemcilerden ve işlevlerin ölçümlerinden elde edilir. Geri bildirimlerde öne çıkmalara, aldatmalara ve tutarlılığa dikkat edilmelidir. Geri bildirimlerin tümü farklı kanallardan sorgulanmalıdır.

Süreçlerde, işlevlerdeki tüm ilerlemeler izlenmelidir. Bulunulan yer ile olmak istenilen yer sürekli karşılaştırılmalıdır. Aradaki fark, süreç iyileştirme çalışmalarında odaklanılması gereken kısımlardır. Raporlama, hedeflere ulaşma yolundaki gelişmeleri takip açısından çok önemlidir. Bunun dışında paydaşlar, denetciler ve danışmanlar tarafından yapılacak değerlendirmeler de önemli bilgiler sağlayacaktır.

5. Yazılım Geliştirme Süreç Modelleri

Yazılım Geliştirme Süreç Modelleri, Yazılım Yaşam Döngüsünde belirtilen süreçlerin hangi düzen ya da sırada, nasıl uygulanacağını tanımlar. Yazılım geliştirmenin, bahsedilen zorluklarıyla baş edebilmek için, geliştirmeyi sistematik hale getirmeyi hedefleyen çeşitli süreç modelleri ortaya çıkmıştır. **Bu modellerin temel hedefi;** proje başarısı için, **Yazılım Geliştirme Yaşam Döngüsü** ("Software Development Life Cycle") boyunca izlenmesi önerilen mühendislik süreçlerini tanımlamaktır.

Modellerin ortaya çıkmasında, ilgili dönemin donanım ve yazılım teknolojileri ile sektör ihtiyaçları önemli rol oynamıştır. Yazılım Geliştirme Süreç Modelleri, süreçlerin içsel ayrıntıları ya da süreçler arası ilişkilerle ilgilenmez. Özetle yazılım üretim işinin genel yapılma düzenine ilişkin rehberler olarak kullanılabilir.

Örnek:

- Geleneksel modeller (Çaglayan ("waterfall") , evrimsel, döngüsel …vb.)
- Çevik ("Agile") Modeller (Uçdeger ("extreme") programlama modeli – XP)

Düzenleyici Süreç Modelleri:

- Kodla ve Düzelt (Code and Fix)
- Gelişigüzel Model
- Barok Modeli
- Çağlayan/Şelale Modeli (Waterfall Model)
- V Modeli (V-shaped Model)
- Prototipleme
- Helezonik Model (Spiral Model)
- Evrimsel Geliştirme Modeli, (Evolutionary Development)
- Artırımsal Geliştirme Modeli, (Incremental Development)
- Araştırma Tabanlı Model, (Resource Based Model)
- Formal Sistem Geliştirme, (Formal System Development)
- Bileşen Tabanlı Geliştirme, (Component Based Development)

Çevik Yazılım Süreci

- Uçdeger Programlama ("Extreme Programming – XP")
- Scrum
- Özellik Güdümlü Gelistirme ("Feature-Driven Development – FDD")
- Çevik Tümlesik Süreç ("Agile Unified Process – AUP")

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 "NATO

Bilim Komitesi" tarafından 1968 yılında Almanya'da "Yazılım Mühendisliği Konferansı" düzenlendi. Bu konferansta üzerinde durulan temel konular yazılım mühendisliği kavramının tanımlanması ve yazılım geliştirme süreçlerinde uygulanması gereken standartların tespit edilmesidir. 1970 yılında Dr. Winston Royce tarafından yayınlanan bir bildiride "Şelale Modeli" tanımlandı. Şelale modeli zaman içinde bugün bildiğimiz şeklini aldı ve yazılım geliştirme metodolojilerinin temeli haline geldi.

5.1. Şelale (Waterfall) Modeli

Ardışıl Model - Şelale Modeli (Sequential / Waterfall), yazılımın tümü teslim edildikten sonra karşılaşılan hataların düzeltilmesi prensibine dayanmaktadır.

Şelale modeli, yazılım projelerinde uygulanan faaliyetlerin ardışık süreçler halinde icra edildiği, yazılım mühendisliğinin en eski ve temel modelidir. Günümüzde hükümetler ve büyük şirketler tarafından her türlü projenin yönetim standardı olarak kabul görmektedir.

Şelale modeli,

- **Sistem ve Yazılım Gereksinimleri,**
- **Analiz,**
- **Tasarım,**
- **Kodlama,**
- **Test ve Entegrasyon,**
- **Sınama, İdame ve Bakım"** adımlarından oluşur.

Bir adımın tamamlanmasından sonra diğerine geçilir. Eksiklikler veya hatalar fark edilirse önceki adımlara geri dönülür. Adımları geride bıraktıkça, ilerleyen aşamalarda karşılaşılan hataların düzeltilmesi üstel olarak zorlaşmaktadır. Son ürünün eldesi uzun süreceğinden müşteri sabırlı olmalıdır. Bir çok müşteri, gereksinimlerini eksiksiz ve kesin belirtmekte zorlanmaktadır. Rutin projelerde uygudur.

Her sürecin sonunda dokümantasyondan oluşan bir çıktı elde edilir ve her çıktı kendini takip eden sürecin girdisini oluşturur. Çıktı-girdi ilişkisi nedeniyle yaşanacak olası değişimlerin projeye maliyeti yüksek olacağı için süreçler arasında atlama veya büyük çapta geri dönüşlere izin verilmez. Şelale modelinin uygulandığı projelerde süreç bitişleri net bir şekilde tanımlanmıştır ve her süreç değişiminde katı incelemeler, yoğun dokümantasyon ve yönetim onayı gibi kontrollerle proje yoğun bir denetim altında icra edilir. Bu nedenle uygulanması katı bir disiplin gerektirir. Bu model, "Önden büyük tasarım" felsefesini benimser ve kodlama sürecinden önce büyük bir analiz ve tasarım yapılır. Sistem ve yazılım gereksinimleri, gerekli her şeyin baştan bilindiği ve değişiklik olmayacağı kabullemeler kabul edilerek en başta belirlenir. Bu nedenle belirsizlik içermeyen tam anlaşılmış ve değişme

ihtimali olmayan projeler için daha uygundur. Çünkü Şelale Modelinin en büyük zayıflığını bu kabullenmeler oluşturmaktadır.

Müşteri, "Gereksinimler" tanımlanırken projenin içindedir, "Analiz, Tasarım ve Kodlama" süreçlerinde görünmez, "Test ve Entegrasyon" sürecinde yeniden ortaya çıkar. Şelale modelinde en çok planlama, zaman çizelgeleri, hedef tarihler, bütçe ve bütün sistemin bir defada kullanıma alınması konuları üzerinde durulur.

Şelale modelinin temel amacı proje süresince değişikliklere izin verilmeyerek, kapsam, zaman ve kaynaklar gibi faktörleri baştan sabitlemek ve büyük bir risk faktörü olan değişimi etkisiz kılmaktır. Şelale modelinin ortaya çıktığı zaman ve şartlar değerlendirildiğinde, iş süreçleri bilgi sistemlerine günümüzde olduğu kadar bağımlı değildi ve projelerin aceleyle bitirilmesi gerekmiyordu. Teknolojik gelişme ve iş yaşamındaki değişim günümüzde olduğu kadar hızlı değildi; bu yüzden projeler günümüzde olduğu kadar hızlı bir tempoyla ilerlemiyordu. Teknoloji ve kaynak kısıtlarından dolayı bilgi sistemleri donanım merkezliydi ve yazılımın ana maksadı, depolama kapasitesi ve işlemci gücü gibi kısıtlı donanımsal kaynakları en optimum şekilde kullanabilmekti.

Şelale modeli daha çok bir risk yönetim metodu olarak görülmektedir. Risk yönetimi için gerekli görülen hususlar şunlardır:

- Planlama ve bütçelemenin doğru yapılabilmesi için bütün gereksinimler önceden tanımlanmalı.
- Gereksinimleri önceden tanımlamak ve değişiklik olmayacak şekilde sabitlemek için işin kapsama aşımdırması önlenmelidir.
- Gelecekte referans oluşturması için bütün tasarım ve karar verme süreçleri belgelenir.
- Süreçler arası yoğun bilgi paylaşımı, bütün paydaşların süreçten haberdar olmasının yanında doğabilecek problemlerin de erken tespit edilmesini sağlar.
- Mimarinin uygun olduğundan emin olunabilmesi için, tasarım faaliyetleri başlamadan önce bütün gereksinimler belgelenir.
- Tüm sistemin kabiliyetlerinin değerlendirildiğinden emin olunabilmesi için test işlemlerinden önce bütün kodlama işlemleri tamamlanır.
- Sistem uygulamaya koyulmadan önce bütün gereksinimler karşılanır.

Şelale Modelin Faydaları (Benefits):

- Projenin başlangıcında gereksinimler, açık, anlaşılır ve tam olarak tanımlanır.
- Doğru zaman planlama ve maliyet tahmini yapılarak projedeki belirsizlikler azaltılır.
- Eskiden çok kullanılan bir metot olduğu için genelde teoride iyi bilinir.
- 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 süreç başlanmadan önce bir önceki süreç tamamlandığı için proje ilerleyişinin takibi kolaydır. Ayrıca eş zamanlı ilerleyen süreçler olmaması kaynak takibini de kolaylaştırır.
- Dokümantasyonun devamlı bir faaliyet olması, projenin mevcut halinin değerlendirilebilmesine imkân verir.
- Şelale modeli disiplinli ve istikrarlı bir modeldir ve proje elemanlarına projenin iyi bir şekilde yönetildiği ve her şeyin kontrol altında olduğu hissiyatını verir.
- 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.
- Hazırlanan detaylı ve kapsamlı dokümantasyon bir iletişim aracı olarak kullanılır ve özellikle büyük projelerde ve bütün elemanların aynı ortamda bulunmadığı projelerde yüz yüze iletişim ihtiyacını azaltır.

Şelale Modelini Kısıtlamaları (Limitations):

- Değişiklik yapma, geliştirme veya düzeltme maksatlı **geriye dönüşler** şelalede akıntıya karşı yüzmeye benzetilir, zor ve maliyetlidir. Bu yüzden değişikliklere anlık durum tespitinde cevap veremez ve süreç içerisinde gelen istekler çoğunlukla kabul edilemez.
- Herhangi bir süreçte tespit edilmemiş bir olumsuzluk sonraki bütün süreçleri etkiler.
- Yapısal özelliği ve sıkı kontrolleri nedeniyle esnek olmayan, yavaş ve hantal bir yapısı vardır.
- Gereksinimlerin proje başında tam ve anlaşılır bir şekilde tanımlanması gerçekçi değildir. Müşterilerin ihtiyaçlarını tam ve doğru ifade edememesi, sonradan oluşan istekler proje sonunda eksik veya yanlış bir ürün elde edilmesine neden olur.
- Geliştiricilerin gereksinimleri müşteri veya kullanıcılarla yüz yüze görüşerek anlamaları yerine analiz ve tasarım süreçlerinde 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 dokümantasyon zaman gerektiren bir faaliyettir ve genel olarak geliştiriciler tarafından gereksiz bir yük olarak görülür.
- Yazılı sistem tanımlamalarının kullanıcı veya müşteriler tarafından okunarak tam olarak anlaşılması ve uygulanması zordur.
- Bir sürecin tamamlanmadan sonraki sürecin başlamaması nedeniyle, tüm proje elemanlarını ilgilendirmeyen gecikmeler dahil bütün proje elemanları için zaman israfına neden olur.
- Yüksek yönetim giderleri küçük projeler ve küçük takımlar için gereksiz maliyetlerdir.

- **Yazılım geliştirme faaliyetleri yaratıcılık gerektiren faaliyetlerdir ve şelale modelinin yüksek disiplinli yaklaşımı yaratıcılığı olumsuz etkiler.**

5.2. Çevik (Agile) Süreçler

Çevik yazılım yönetim süreçleri kendi kendini organize eden takımlar tarafından geliştirilir. Son derece işbirlikçi kurumsal bir anlayışla uygun maliyetli ve paydaşların değişen ihtiyaçlarını karşılayan yüksek kalitede çözümler üreten yinelemeli ve artımlı (evrimsel) bir yaklaşımdır. Çevik yöntemler, geleneksel yöntemlerin yetersiz kaldığı değerlendirilen konular için alternatif çözümler olarak ortaya çıkmıştır.

Süreçlerdeki işlevlerden ziyade paydaşların etkileşimlerine, kapsamlı dokümantasyondan ziyade üretilen koda, sözleşme pazarlıklarından ziyade müşteri ile işbirliğine, bir plana bağlı kalmaktan ziyade değişimi fark etmeye odaklanılır. Değişen gereksinimler, teknik riskler gibi önceden belirlenemeyen durumlara ve yazılım ürününü etkileyebilecek her tür değişikliğe karşı esneklik sağlayan süreçtir.

Yazılım süreç yönetiminde bir ilerleme olmaksızın yalnızca sürekli uyum sağlamak başarı değildir. Çünkü kod yazılımı artımsal gelişir, müşteriye erken ve sık ürün teslimi sorun yaratmaz, başarıının birincil ölçütü çalışan yazılımdır.

Çevik yönetim sürecinde yer alacak ekibin özellikleri:

- Sağlıklı iletişim ortamında, yüz yüze görüşme, en etkili bilgi aktarım yoludur.
- Takım üyeleri çevik yaklaşım hakkında eğitilmelidir.
- Ekip üyelerinin ortak amacı, çalışan yazılım üreterek müşteriye zamanında teslim etmek olmalıdır.
- Ekip üyeleri birbirleriyle (paydaşlar) ve müşteriyle işbirliği içinde olmalıdır.
- Ekip üyeleri karşılıklı saygı ve güven içerisinde olmalıdır.
- Ekip hem teknik, hem de tüm proje hakkında kararlar verebilmelidir.
- Boşuna harcanan çaba yoktur: Çözülen bir sorun gereksizleşse bile, çözüm sürecinde edilen deneyim, kazanılan yetenek ekibe ileri aşamalarda yararlı olacaktır.

Kendi kendini düzenleme:

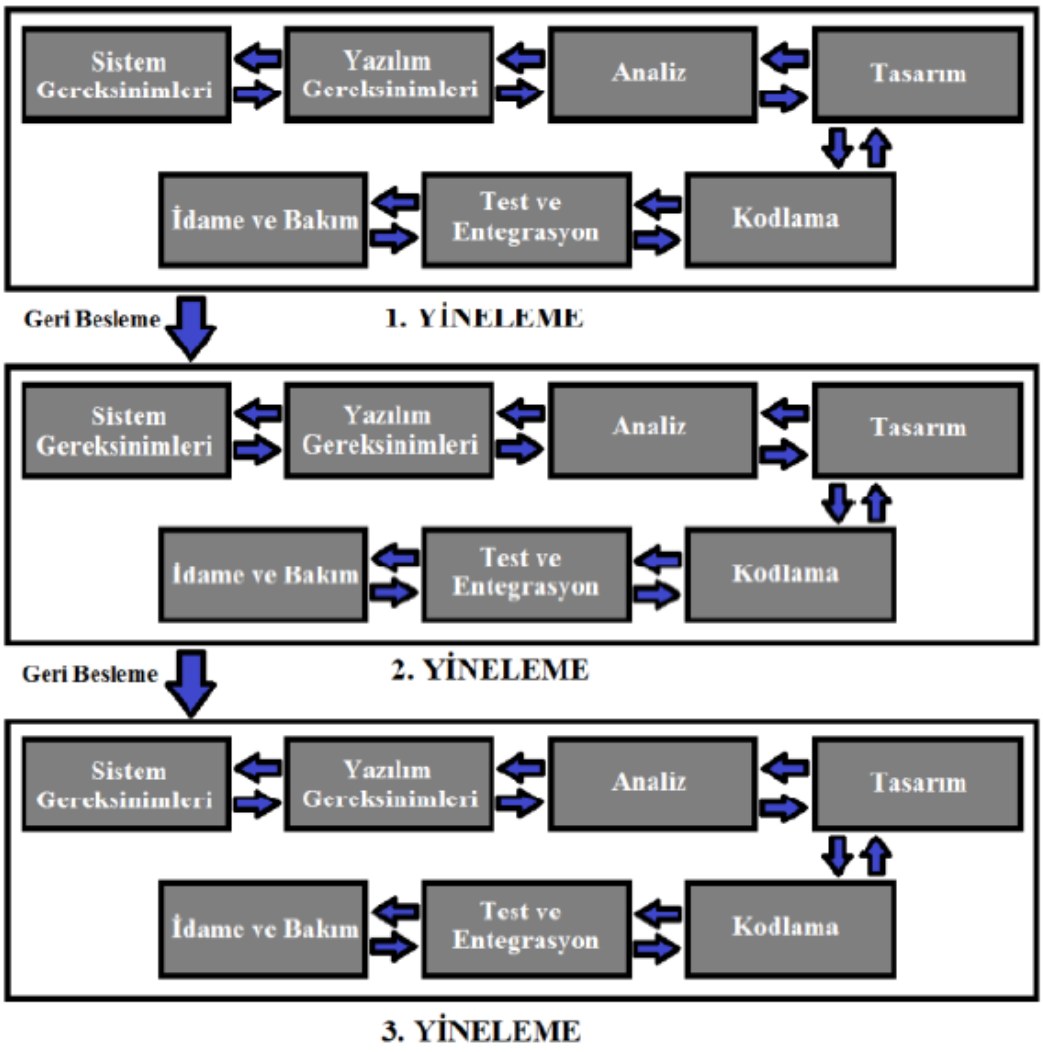
- Ekibin kendisini yapılacak işe göre uyarlaması,
- Ekibin kullanacağı süreci işlevlere uyarlaması,
- Üstünde çalışılan artımsal yazılım işlevlerini kısım kısım teslim etmek için gerekli çalışma zaman planlamasını ekibin kendisinin belirlemesi.

Çevik yöntemleri felsefesinden doğan oniki ilke vardır:

- Yazılımın erken ve belirlenen fazlardaki işlevlere göre teslimi sağlanarak müşteriler memnun edilir.
- Değişen gereksinimler yazılım sürecinin son aşamalarında bile kabul edilir. Çevik süreçler, değişimi müşterinin rekabet avantajı için kullanır.
- Çalışan yazılım, belirlenen zamanlarda düzenli olarak müşteriye sunulur.
- İş süreçlerinin sahipleri ve yazılımcılar proje boyunca birlikte çalışır.
- Projelerin temelinde motive olmuş, odaklı bireyler yer almalıdır. Onlara ihtiyaçları olan ortam ve destek sağlanmalı, işi başaracakları konusunda güven duyulmalıdır.
- Bir yazılım takımında bilgi alışverişinin en verimli ve etkin yöntemi yüz yüze iletişimdir.
- Çalışan yazılım ilerlemenin birincil ölçüsüdür.
- Çevik süreçler sürdürülebilir işlevler geliştirmeyi teşvik etmektedir.
- Teknik alt yapı desteği, mükemmeliyet ve iyi tasarım konusundaki sürekli özen çevikliği artırır.
- Sadelik, yapılmasına gerek olmayan işlerin mümkün olduğunca arttırılması sanatı, olmazsa olmazlardandır.
- En iyi mimariler, gereksinimlerde ve tasarımlarda **kendi kendini örgütleyen takımlardan** ortaya çıkar.
- Takım, nasıl daha etkili ve verimli olabileceği üzerine beyin fırtınası yapar ve davranışlar değişimlere göre ayarlanır ve düzenlenir.

Çevik yöntemlerde uygulanan yinelemeler, önceki yinelemelerde elde edilen tecrübeler ve tespit edilen aksaklıklar doğrultusunda şekillenir. Yapılması gereken görevler, taşıdıkları iş değerine göre önceliklendirilir. Yapılacak işin en iyi nasıl yapılacağını, mevcut kaynaklar ve kısıtlar çerçevesinde proje takımı kendi belirler. Takım belli görevleri belli sürelerde (yineleme süresi içinde) tamamlanır. Yineleme sonunda teslim edilecek ürünü meydana getirmekten sorumlu olan güçlü ve zayıf yanlarıyla takımdır. Bu nedenle takım içi işbirliği önem arz eder. Bu kapsamda çevik yöntemlerin dayandığı temel esaslar, deneysellik, önceliklendirme, kendi kendini örgütleme, zaman planlaması ve iş birliği şeklinde sıralanabilir.

Çevik yöntemlerde yazılım geliştirme faaliyetleri yinelemeli süreçler halinde uygulanır. Bu yinelemeler sonucunda kullanılabilir ürün ortaya çıkarılır. Paydaşların, Müşteri veya kullanıcıların tecrübe geri beslemeleri ışığında ve varsa değişen gereksinimler doğrultusunda geliştirme süreçleri tekrarlanır. Bu yinelemeli süreçler müşterinin tam olarak istediği ürün ortaya çıkana kadar sürdürülür. Çevik yöntemlerde uygulanan süreçler aşağıdaki şekilde gösterilmiştir.



Çevik Yöntemlerin Genel Uygulaması

Faydaları(Benefits)

- Çevik yönteminin felsefesi gereği proje süresince değişikliklerin hoş karşılanması ve hatta teşvik edilmesi nedeniyle ve tekrarlamalı süreç yapısı sayesinde esneklik yüksektir ve değişen şartların yönetilmesi kolaydır.
- 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.
- 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.
- Problemler ve istekler kişiselleştirilemez.
- Çevik yöntemler, uygulanan süreçler arası geçişlere ve geri dönüşlere müsait olduğu için süreç devam ederken tespit edilen geriye dönük aksaklıklar kolaylıkla düzeltilebilir.
- Genel olarak üretilen yazılımın sadece ürün özellikleri dokümantasyona çevrilir ve diğer süreçler hariç tutulur. Hafif dokümantasyon iş yükünü azaltır.

Çevik süreçlerin dezavantajları:

- Uygun olmayan ekiple (eğitimsiz, ekip çalışmasına uygun olmayan, işini kişiselleştiren, kişilik problemleri olanlar) çevik süreç yönetimi çalışamaz.
- Kalabalık ekip veya büyük ölçekli projeler için uygun görülmemektedir. Ekip ve projeler parçalara ayrılırsa, iyi bir yönetim ile başarmak mümkündür.
- Bir dış denetleyicinin dahil olduğu ve ayrıntılı kuralların gerektiği denetlemelerin zorunlu olduğu projelerde yetersiz kalmaktadır.
- Çevik çalışmak disiplinsizlik olarak yorumlanmamalıdır.

Kısıtları (Limitations)

- Çevik yöntemlerde süreçler iletişim merkezli olduğu için proje elemanlarının birbirleriyle veya müşteriyle sağlıksız iletişimi proje gelişimini olumsuz etkiler. Video konferans gibi uzaktan iletişim teknolojileri kullanılmalıdır.
- İsteklerin kesin olması ve değişmemesi gerektiği durumlarda, 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.

- Değişime açık olması nedeniyle projenin kaynak ve kapsam planlaması zordur.
- ”Gerektiği kadar” anlayışı ile hazırlanan dokümantasyonun yetersiz olması durumunda projeye sonradan katılan geliştiriciler ve bakım-idame süreçleri olumsuz etkilenir.
- Takımlar kendi kendini organize ettiği için bireylerin yetenekli, tecrübeli ve çevik süreçlere hâkim olması gerekir. Yeteneksiz, sıkıntılı, ekip çalışmasına hazırlıksız, tecrübesiz elemanlar 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 olmayabilir.
- 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.

Geleneksel şelale modeli ile çevik yöntemler arasında yapılan işin başarı oranları uygulanan metodolojiye göre de farklılık göstermektedir. Şelale modeli ile çevik yöntemler aynı amaç doğrultusunda ortaya çıkmış yaklaşımlardır. Her iki yaklaşımda da planlama önemli bir yer tutar. Şelale modelinde bütün süreci kapsayan uzun vadeli ve detaylı bir planlama yapılırken, çevik yöntemlerde değişen şartlara cevap verebilmek için uzun vadeli planlamadan ziyade kısa vadeli ve daha detaylı planlamalar yapılır. Bütün süreç sürüm planlarına, sürüm planları yineleme planlarına, yineleme planları ise günlük planlara bölünür. Böylece daha detaylı hedefler belirlenebilir. Her iki yaklaşımda da süreç içinde detaylı incelemelere önem verilir. Şelale modelinde bu incelemeler süreçler arası geçişte yapılan detaylı kontroller ve dokümantasyon vasıtasıyla yapılırken, çevik yöntemlerde yinelemeler sonucu üretilen çalışan ürünler ve müşterinin bütün sürecin içinde bulunması ile detaylı bir inceleme sağlanmış olur. İki metodolojide de yapılan faaliyetler ortaktır. Bu faaliyetlerin icra şekilleri farklıdır. Şelale modelinde bu faaliyetler her gereksinime uygulanan süreçler olarak karşımıza çıkar ve ana sürecin alt bileşenleridir. Çevik yöntemlerde ise her yinelemenin uygulandığı faaliyetler olarak karşımıza çıkarlar ve ana sürecin alt bileşeni yinelemeleri oluşturan faaliyetlerdir.

Şelale modeli, proje başlangıcından itibaren hem üretim hem de kabul risklerini barındırmaktadır. Süreç içerisinde uygulanan önlemler ile risk bir miktar azalsa da kabul aşamasına kadar ana hatlarıyla devam etmektedir. Çevik yöntemlerde ise tamamlanan her

yinelemede üretim riskleri, her teslimde ise müşteri katılımı sağlandığı için kabul riskleri azalmaktadır.

Şelale modelinde gereksinimlerden oluşan paket proje sürecinde süreçler üzerinde ilerlerken çevik yöntemlerde süreçlerdan oluşan paket yinelemeler üzerinde ilerler. Projenin başında tanımlanan gereksinimler ve önceliklere göre teslimat yapılması nedeniyle şelale modelinde devamlı geriye doğru bir bakış vardır. Çevik yöntemlerde ise evrilen gereksinim ve önceliklere göre teslimat yapılması nedeniyle devamlı ileriye doğru bir bakış vardır. Yapısal süreçlere dayalı olması nedeniyle şelale modeli daha çok bir mühendislik yaklaşımı sunarken kişilere ve iletişime dayalı olması nedeniyle çevik yöntemlerde yaratıcılık ön plandadır.
Çalışmanın bundan sonraki bölümünde; Tablo 2"de belirtilen CHAOS raporları başarı faktörleri, literatür araştırmaları ve yazılım projelerinde edinilen kişisel tecrübeler ışığında proje başarısına etki ettiği değerlendirilen konularda, şelale modeli ve çevik yöntemler için karşılaştırma ve değerlendirmeler sunulacaktır.

Çevik Süreç Örnekleri:Aşırı Programlama (XP: ExtremeProgramming), Scrum
Başlıca çevik metodolojisi Scrum olmakla birlikte XP ve Kanban da kullanılmaktadır. Scrum yönteminin Kanban ve XP ile beraber kullanıldığı ve organizasyonların kendi ihtiyaçlarına göre uyarladığı karma metodolojiler de çevik yöntemlerin başlıca kullanım şekillerindendir.

Aşırı Programlama (XP)

Adımlar: Planlama –Tasarım –Kodlama –Sınama Artımsal Ürün

Planlama:

- Müşteri, kullanıcı öyküleri oluşturur.Müşteri, öyküleriönemine göre derecelendirir. Gerçeklenemeyecek öykülervarsa, ekip müşteriden bunları alt öykülerebölmesini ister.
- Ekip ve kullanıcı, öykülerin oransal olarak ürünü nasıl ekleneceğine karar verir:Ya önce yüksek riskli öyküler gerçeklenir,Ya da önce yüksek öncelikli öyküler gerçeklenir. Her olasılıkta tüm öyküler kısa sürede (birkaç hafta) gerçeklenmelidir.
- İlk oransal ürün projenin hızını ölçme amacıyla değerlendirilir:Eldeki artımın hızına göre sonraki artımların teslim tarihleri belirlenir. Aşırı sözler verildiği ortaya çıkarsa artımsal ürünlerin içeriği de yeniden kararlaştırılabilir.
- Süreç ilerledikçe müşteri yeni öyküler ekleyebilir, eski öykülerin önceliğini değiştirebilir, öyküleri farklı şekillerde bölüp birleştirebilir, bazı öykülerden vazgeçebilir.Bu durumda ekip kalan artımları ve iş planlarını uygun biçimde değiştirir.

Tasarım:

- Basit tasarım karmaşık gösterimden üstündür. (KISS: Keep It Simple, Stupid!)
- (Class-Resposibility-Collaboration) kartları ile yazılımın sınıf düzeyinde incelenmesi.
- Karmaşık bir tasarımdan kaçınılamazsa işlevsel bir ön gerçekleme yapılır (Spike solution).
- Refactoring teşvik edilir.
- CRC kartları ve ön gerçeklemeler üretilir (başka ürün yok).

Kodlama:

- Sınamalar hazırlanır. Programcı tarafından yapılan, sınıfların (NYP'de; yapısal'da fonksiyonlar, vb.'lerin) temel işlevselliklerini sınama amaçlı kod. Sadece sınavı geçmeye yarayan kod yazılır (KISS).
- Çift kişi ile kodlama: Bir programcı eldeki sorunu çözerken diğeri çözümün genel tasarıma uygunluğunu gözetir ve kodlamanın takımın karar verdiği ölçütlere (kalite, vb.) uygunluğunu denetler.

Sınama:Birim sınamalarının otomatik çalıştırılması.Müşterinin artımsal ürünü denemesi.

Scrum

Scrum, 1990'ların başından beri karmaşık ürün geliştirme sürecini yönetmek için kullanılan bir süreç çerçevesidir. Scrum çerçevesi, Scrum Takımları ve takımlarla ilgili rolleri, etkinlikleri, eserleri ve kuralları kapsar. Scrumın temelinde deneysel süreç kontrol teorisi yer alır. Scrum, öngörülebilirliği en iyi seviyeye çıkarmak ve riski kontrol etmek için iterasyonlu ve artımlı bir yaklaşım kullanır.

Scrum Takımı

Scrum Takımı, bir Ürün Sahibi, Geliştirme Takımı ve bir de Scrum Masterdan oluşur.

Geliştirme Takımı, her Yazılım Problem Raporlamaint sonunda "Bitti" tanımına uyan ve yayınlanabilir ürün parçası teslim etmekle, **Ürün Sahibi**, Geliştirme Takımının işini ve ürünün değerini en üst seviyeye çıkarmakla, **Scrum Master,** Scrumın anlaşılmasını ve uygulanmasını temin etmekle sorumludur.

Scrum Etkinlikleri

Bir ay veya daha az zaman sınırı olan, içerisinde "Bitti" durumunda, kullanılabilir ve potansiyel olarak yayınlanabilir bir Ürün Parçasının oluşturulduğu **Yazılım Problem Raporlamaint**, Scrumın kalbidir. Yazılım Problem Raporlamaintte yapılacak iş **Yazılım Problem Raporlamaint Planlama** toplantısında planlanır. **Yazılım Problem Raporlamaint Değerlendirme**, her bir Yazılım Problem Raporlamaintin sonunda Ürün Parçasını görüp kontrol etmek ve gerekiyorsa Ürün İş Listesini uyarlamak için düzenlenir. **Yazılım Problem**

Raporlamaint Retrospektifi, Scrum Takımının kendini gözlemlemesi ve sıradaki Yazılım Problem Raporlamaintte yapacağı iyileştirmelere ilişkin bir plan oluşturması için bir fırsattır.

Scrum Eserleri

Ürün İş Listesi, üründe ihtiyaç duyulan her şeyin sıralandığı bir liste, yani gereksinimler kaynağıdır. **Yazılım Problem Raporlamaint İş Listesi**, Yazılım Problem Raporlamaint için seçilen Ürün İş Listesi kalemlerini ve Yazılım Problem Raporlamaint hedefine ulaşma planını içerir. **Ürün Parçası**, bir Yazılım Problem Raporlamaint boyunca tamamlanan Ürün İş Listesi kalemlerinin ve tüm geçmiş Yazılım Problem Raporlamaintlerin Ürün Parçalarının değerlerinin toplamıdır.

Diğer Modeller

Ön Ürün Modeli (Prototip modeli)

Müşteriyi dinle –Ön ürün oluştur –Müşteri ön ürünü dener adımlarından oluşur. Kullanıcı gereksinimlerinin daha iyi elde edilir. Kullanıcı erkenden ürünü değerlendirmeye başlayabilir.

Eksik ürün zaman ve maliyet kısıtlamaları nedeniyle olgunlaşmadan canlı kullanıma alınabilmektedir.Prototip oluşturma başlı başına bir model olarak kullanmamalı, daha olgun bir modelin analiz aşamasında kullanılacak bir araç olarak ele alınmalıdır.

Hızlı uygulama geliştirme (RAD: Rapid Application Development)

Kısa geliştirme çevrimleri üzerinde duran artımsal bir modeldir. Koşulları vardır: Uygulamanın tahmini ortalama sürelere ayrılabilmesi, yeterli sayıda bölümün eşzamanlı ilerlemesinin sağlanabilmesi, yazılımın bileşenlerden kurulabilmesi.

Bu sürece uygun yazılım projelerinde verimlilik artar. Ancak büyük ölçekli çalışmalarda yeterli sayıda bölümü eşzamanlı ilerletebilecek sayıda çalışan bulunamayabilir. Çalışanlar hıza uyum sağlamayabilir.Yüksek teknik risklere uygun değildir.Sonuç olarak prototip geliştirmede kullanılması veya ana fikirlerinin diğer süreçlere uygulanması yerinde olacaktır.

Bileşen tabanlı (Component Based) UYGULAMA GELİŞTİRME

Uygulamanın hazır yazılım bileşenlerinden oluşturulmasını öngörür.

Aşamaları:

- Konu alanı mühendisliği (Domain Engineering)
- Aday bileşenlerin sınıflandırılması ve seçilmesi (Qualification)
- Seçilen bileşenlerin kendi yazılımımıza uyarlanması (Adaptation)
- Bileşenlerin bir araya getirilmesi (Composition)

Artılar:Yeniden kullanımın özendirilmesi (azalan giderler?)

Eksiler:

- Uygun bileşenlerin bulunması gerekliliği (her zaman bulunmaz)
- Bileşenlerin uyarlanması gerekliliği (göründüğü kadar kolay olmayabilir)

Sonuç:Özellikle hızlı uygulama geliştirme olmak üzere, ana fikirleri çeşitli süreçlere uygulanabilir.

Artımsal / Yinelemeli Modeller

Artımsal / Yinelemeli Modeller (Incremental / Iterative)
Adımlar: Analiz –Tasarım –Kodlama –Sınama –Bakım
Gereksinimler önemlerine ve birbirine bağımlılıklarına göre sıralanarak her yinelemede bunların bir kısmı tamamlanır.

Artılar:

- Ön ürün modeli ve ardışıl modelin güçlü yönlerini kendinde toplayarak dezavantajlarını geride bırakmıştır.
- Nesneye yönelik programlama metodolojisi ile uyum içerisindedir.

Eksiler: Yazılımın küçük artımlarına fazla yoğunlaşmak, sistemin geneline bakıldığında kolayca görülebilecek sorunların gözden kaçmasına neden olabilir.

Sonuçlar:Sistemin genelini göz ardı etmemek şartıyla güçlü bir modeldir.

Örnekler: Spiral Model ve Kazan-Kazan Modeli

SARMAL (Spiral) MODEL

Müşteri ile İletişim: Gereksinimlerin Belirlenmesi
Planlama: Kaynaklar, zamanlama, yapılacaklar, vb.
Risk Analizi: Teknik, mali ve politik riskler
Çözümleme ve Tasarım
Gerçekleme ve Kurulum: Kullanıcı eğitimi, dokümantasyon, vb. dahil.
Müşteri Tarafından Ürünün Değerlendirilmesi

Kazan-Kazan Modeli (WINWIN Model)

İlgililerin (paydaşlar) Belirlenmesi
Kazanma Durumlarının Belirlenmesi
Uzlaşma ve Seçeneklerin Belirlenmesi
Seçeneklerin Değerlendirilmesi ve Risklerin Çözülmesi
Gerçekleme ve Kurulum
İlgililer Tarafından Değerlendirmeler

5.3. Yazılım Modellerinin Karşılaştırılması

Yönetim ve Roller:
Merkezi ve hiyerarşik yapılı yönetim sistemleri, şelale modeli ile uyumludur. Liderlik, dayanışma ve çalışanların işlerini kolaylaştırıcı anlayışta yönetim tarzı ise çevik yöntemler için uygun ortamlardır. Roller açısından da iki metodoloji arasındaki en temel fark yöneticilerdir. Çevik yöntemlerde takımı motive eden, engelleri kaldıran, süreçleri kolaylaştıran bir koçluk ortaya koyulurken, şelale modelinde klasik komuta-kontrol yönetim tarzı uygulanır. Kendi kendini organize bağlamında proje yönetimi faaliyetleri liderler, müşteriler ve takım elemanları arasında paylaşılır.

Görünürlük:
Şelale modelinde proje görünürlüğü uygulanan süreçler arasında yapılan kontroller ve dokümantasyonla sağlanırken, çevik yöntemlerde yinelemeler sonunda ortaya çıkarılan çalışır ürünler ve ürün iş listeleri üzerinden sağlanır. Çevik yöntemler bu kapsamda müşteriden doğrudan çalışan ürünün onayını alarak proje ilerlemesini gerçekleştirdiği için şelale modeline göre avantaj sağlar. Çevik yöntemlerin görünürlük olarak avantajlı olduğu projeler, çalışan ürünün kullanıcı ara yüzlü ürün olduğu projelerdir. Kullanıcı tarafından değerlendirilemeyen ya da fonksiyonel olarak küçük parçalara ayrılamayan projelerde dokümantasyon daha iyi bir görünürlük sağlayacağı için şelale modeli daha uygundur.

Karmaşıklık:
Üretilecek yazılımın karmaşıklık seviyesi arttıkça analiz ve yönetim ihtiyacı da artar. Örneğin teknik ve yönetsel açıdan en karmaşık projeler olan gerçek zamanlı savunma sistemleri için kodlama aşamasına kadar detaylı analizler yapılmış olmalıdır. Bu şartlarda şelale modeli, plan tabanlı bir yaklaşım olduğu için en iyi seçenektir. Çevik yöntemler en erken zamanda müşteri için değer üretmeyi amaçlar ve eldeki mevcut girdilerle kodlamaya geçilerek yinelemeler içerisinde durumun olgunlaşması amaçlanır. Bu nedenle çevik yöntemler daha az analiz gerektiren nispeten daha az karmaşık sistemler için daha uygundur. Detaylı analiz ve planlama şelale modelinin çevik yöntemlere göre avantajı iken, yineleme sonlarında yapılan çalışır ürün teslimleri vasıtasıyla yapılabilen gözden geçirme, alınan geri besleme ve değişim yapabilme imkânı, proje sürecinde karmaşıklık ile başa çıkabilme açısından çevik yöntemlerin şelale modeline göre avantajıdır.

Büyüklük:
Proje büyüklüğü ile ilgili genel kanı, büyük projelerin detaylı dokümantasyon ile daha iyi yönetilebileceği, küçük projelerin ise katılımcılar arasındaki doğrudan iletişim sayesinde daha etkin sürdürülebileceği, bu nedenle büyük projelerin şelale, küçük projelerin ise çevik yöntemler için daha uygun olduğu yönündedir. Proje büyüdükçe çalışan sayısı ve bütçe artar. Bu da daha çok koordinasyon gerektirir. Şelale modeli, kapsamlı planlama,

dokümantasyon ve büyük takımlar için daha iyi iletişim ve koordinasyon süreçleri sunarak bu ortamı destekler.

Proje büyüdükçe çevik yöntemlerin maruz kaldığı dezavantajlara karşı tedbirler alınmakta ve büyük ölçekli projelerde kullanılmak üzere özelleşmiş geniş kapsamlı çevik metodolojiler ortaya çıkmaktadır. Geniş kapsamlı çevik yöntemlerin en önde gelenleri Disciplined Agile Delivery (DAD), Large-Scale Scrum (LeSS), ve Scaled Agile Framework (SAFe) metodolojileridir.

Altyüklenici Kullanımı:
Altyüklenici kullanılacak durumlarda, yapılması istenen ürün veya ürün parçasının detaylı tanımlanması şarttır. Altyüklenici kendisinden istenen ürüne göre bir maliyet, zaman ve efor tahmini hazırlayarak bir teklif verir ve sözleşme yapılır. Maliyetleri arttıracağı için ihtiyaçlarda doğabilecek değişiklikler kabul edilmez. Bir tedbir alınmadığı müddetçe ihtiyaç sahibi müşteri ve proje takımı ile alt yüklenici arasında herhangi bir iletişim olmaz. İletişim olmadan müşteri etkileşimi ve proje takımı ile dayanışmadan söz edilemez. Bu nedenlerle çevik yöntemler, altyüklenici kullanımına uygun değildir. Tanımlanmış isterler ve zaman kısıtları nedeniyle şelale modeli alt yüklenici kullanımına daha müsaittir.

Müşteriler(Customers)
Müşteri katılımının istenen seviyede sağlanabildiği durumlarda çevik yöntemler proje sonucunda elde edilecek yazılımın nitelikleri açısından şelale modeline göre daha avantajlıdır. Gereksinimlerin ve iş süreçlerinin sağlıklı bir şekilde tanımlanması, ara teslimatlar ve geri beslemeler sayesinde elde edilen ürünün tam olarak istenen ürün olması sağlanabilir. Bunun yanında proje sürecine müşteri katılımının sağlanabilmesi için müşteri açısından istek, zaman ve bilgi gerekir. Proje elemanlarıyla birlikte proje süresince çalışacak eleman bulundurmak müşteri için külfetli olabilir ve istenmeyebilir. Geleneksel proje yönetimine alışkın müşteriler proje başında ihtiyaçlarını tanımlayıp bir daha zaman ayırmak istemeyebilir çünkü müşterilerin kendi günlük işleri her zaman daha önemlidir. Aynı yerde ve zaman diliminde olmayan müşteriler ile yüz yüze iletişim kurulamaması faaliyetleri aksatabilir. Müşteri adına projeye katılan personel, iş süreçlerine hâkim olmayabilir. Böyle durumlarda gereksinimlerin sağlıklı tanımlanamaması ve teslim edilen ürünlerden geri besleme alınamaması muhtemeldir. Bu gibi nedenlerle müşteri katılımının sağlıklı olarak sağlanamayacağı ortamlar açısından şelale modeli daha uygundur çünkü çevik yöntemlerin en önemli dayanaklarından biri müşteri katılımı iken şelale modelinde proje başlangıcından sonra teslime kadar bir müşteri katılımı beklenmez.

5.4. Organizasyon Yapısı (Organizational Structure)

Organizasyon çeşitleri,

- Girişimci Organizasyon,
- Makine Organizasyon,
- Profesyonel Organizasyon,
- Farklılaşmış Organizasyon,
- Yenilikçi Organizasyon,
- Görev Organizasyonu ve
- Politik Organizasyon olarak yedi ana grup altında toplanır.

Yazılım geliştirme organizasyonları genel olarak girişimci, farklılaşmış ve yenilikçi organizasyonlardır fakat makine ve profesyonel organizasyonlarda da yazılım projeleri yürütülebilir. Görev organizasyonu ve politik organizasyonlar yazılım proje yönetimi açısından karşılaşılacak organizasyon tipleri değildir Bu nedenle ilk beş organizasyon ele alınacaktır.

Girişimci Organizasyon:
Bu tip organizasyonlar esnektir ve düşük hiyerarşiye sahiptir. Hiyerarşi, yetki, sorumlulukların belirlenmesidir. Basit ve dinamik bir ortamları vardır. Genellikle kuruluş aşamasındadırlar. Çoğu girişimci organizasyon yüksek derecede tepkiseldir ve görev anlayışları yüksek seviyededir. Çoğu kez organizasyonun başı çalışanlarla birebir iletişim halindedir. En büyük olumsuzlukları dengesiz ve sağlamlaşmamış iş ortamlarıdır. İletişim ortamının müsait ve yapılarının esnek olması nedeniyle ve iş süreçlerinin değişen şartlara uyum sağlaması gerektiğinden bu tip organizasyonlarda çevik yöntemlerin yinelemeli ve değişime açık yaklaşımlarının uygulanması şelale modeline göre daha uygundur fakat genellikle yeni oluşumlar oldukları için yetenekli ve tecrübeli çalışanları istihdam etme konusunda sıkıntı yaşayabilirler.

Makine Organizasyon:
Bu tip organizasyonlar fabrikalar gibi hiyerarşik ve olgunlaşmış yapılardır ve değişime direnirler. Durağan ve stabil bir ortamları vardır. Merkezi bir bürokrasi ve formel süreçler işler. Çalışanlar fonksiyonel gruplara ayrılmıştır ve genelde hep aynı işi yaparlar. En büyük olumsuzlukları kontrol saplantısı nedeniyle çalışanlarla ilgili problemlere açık olmaları ve değişen şartlara uyum sağlayamamalarıdır. Bu tip organizasyonlar her yönüyle şelale modeli için biçilmiş kaftandır ve çevik yöntemlerin uygulanması neredeyse imkansızdır.

Profesyonel Organizasyon:
Bu tip organizasyonlar bürokratik fakat merkezi olmayan, hastane, üniversite benzeri organizasyonlardır. Otonom olarak çalışan profesyonel personel gruplarından oluşurlar. Genel stratejileri stabildir fakat gerekli spesifik detaylar sık sık değişebilir. Çalışanlar işlerini

iyi şekilde bilirler ve yönetim kademesi iş detaylarına karışmaz. Profesyonel organizasyonlarda koordinasyon konusunda problemler yaşanabilir. Şelale modelinin hiyerarşik komuta-kontrol yapısı bu tip organizasyonlar için uygun değildir. Çalışanların profesyonel olması ve organizasyonun kişisel otonomiye önem vermesinden dolayı çevik yöntemler bu tip organizasyonlar için daha uygun yaklaşımlardır.

Farklılaşmış Organizasyonlar:
Farklılaşmış organizasyonlar, bünyelerinde yapılan işin farklılığına göre oluşturulmuş şirket benzeri birimler bulunduran, bu birimlerin insan kaynakları, finans, hukuk gibi ortak fonksiyonlarını da kendi çatısı altında tutan holding benzeri yapılardır. Bu tip organizasyonlar için hiyerarşik yapı ve uzmanlık alanlarındaki farklılıklar nedeniyle şelale modeli daha uygundur. Her birim kendisiyle ilgili faaliyeti icra edecek ve biten işi diğer birime iletecektir. Çevik yöntemlerde gerekli iletişim ve koordinasyon ortamının bu tip dağıtık organizasyonlarda sağlanması zordur. Uzmanlık alanlarına göre personel ayrılarak matriks yapı benzeri proje grupları oluşturulması halinde çevik yöntemler uygulanabilir.

Yenilikçi Organizasyon:
Bu tip organizasyonlar merkezi olmayan ve hiyerarşi seviyesi düşük organizasyonlardır. Fonksiyonel uzmanlardan oluşur ve karşılıklı koordinasyona dayanır. Kimin ne görev yapacağı çok net belirlenmemiştir. Bir masanın etrafında bir problemi çözmek için toplanmış uzmanlar topluluğuna benzetilebilir. Tanımından da anlaşılacağı üzere uzmanlık, koordinasyon ve otonom yapısı nedeniyle çevik yöntemlerin felsefesi ile birebir örtüşen bir organizasyon tipidir. Komuta kontrol yönetim sistemine uymaması nedeniyle şelale modeli için uygun değildir.

5.5. Proje Paydaşları (Project Members)

Roller:

Şelale modeli ile çevik yöntemlerdeki roller başta yönetici rolleri olmak üzere farklıdır. Şelale modelinde proje elemanları, temel yazılım geliştirme ve mühendislik faaliyetlerini geliştirici ve test mühendisi gibi belirlenmiş uzmanlık alan adları altında gerçekleştirirler. Gereksinimlerin toplanması, netleştirilmesi, efor hesaplamaları gibi yönetim faaliyetleri ise proje yöneticileri veya iş analistleri tarafından yapılır. İşler, proje yöneticileri tarafından proje elemanlarına paylaştırılır. Çevik yöntemlerde ise yönetim faaliyetleri lider rollerdeki personelle birlikte kendi kendini organize eden takımların da katılımı ile icra edilir. Otonom takımlar, yazılım geliştirmenin geleneksel sorumluluklarının yanında hesaplamalar, planlama, görev dağılımı, projenin izlenmesi gibi proje yönetim faaliyetlerini de yürütürler. Takımların bu serbestliğinin yanında proje yönetim faaliyetleri ve karar verme süreçlerinin içine girişi, karar verme otoritesini operasyonel seviyeye indirerek süreçlerde hızlanma ve problem çözümlerinde isabet kazandırır.

Tecrübe ve Yetenek:

Çevik yöntemler, kendi kendini organize eden takımlara dayalı olması ve nispeten daha az detaylı tasarımı koda çevirmesi nedeniyle tecrübeli ve yetenekli yazılımcılara ihtiyaç duyar. Takımın kendi kendini ve projenin verimliliğini değerlendirebilmesi gerekir. Bu nedenle takım elemanları bir karşılaştırma yapabilmek için geçmiş projelerde başarı ve başarısızlığı tecrübe etmiş olmalıdır. Proje elemanları tecrübesiz ise otonom bir ortamda kontrolün kaybedilmemesi için plan tabanlı bir yaklaşımla detaylı bir şekilde hazırlanan tasarımın kodlanması daha uygun bir hal tarzıdır.

Takım büyüklüğü:

Takımlar büyüdükçe elemanların iletişim etkinliği azalmaktadır. Ayrıca büyük takımlarda, kendi kendini yönetme becerisi zayıflarken grubun yönetim ihtiyacı doğar. Bu nedenle grup büyüdükçe çevik yöntemler gibi iletişim ve koordinasyona dayalı metodolojilerin verimliliği düşecektir. Şelale modelinde yerleşmiş bir yönetim anlayışı vardır ve yapılan katı iş tanımları nedeniyle iletişim daha az öneme sahiptir. Ayrıca süreç gereği üretilen detaylı dokümantasyon iyi bir iletişim aracıdır. Bu nedenlerle büyük takımlar için şelale modeli daha uygun bir yaklaşımdır.

Alan Bilgisi:

Şelale modelinde proje başlangıcında detaylı gereksinim analizi ve tasarım çalışması yapıldığı için yazılım geliştirme ekibinde çalışan personelin yazılımın kullanılacağı iş alanı ile ilgili bilgisi olmasına gerek yoktur. Geliştirme sürecinde kullanılacak girdiler önceki süreçlerin sonucunda elde edilen çıktılar olduğu için yapılmış analiz ve tasarıma göre kod yazılır. Çevik yöntemlerde ise müşterinin ihtiyacı olan iş süreçlerinin takım tarafından bilinmesine ihtiyaç vardır. Gerek detaylı analiz ve tasarım eksikliği gerekse değişimlere cevap verme zorunluluğu nedeniyle alan bilgisi eksikliği çevik yöntemlerde problemlere neden olabilir.

Müşteri katılımı az olduğunda alan bilgisi olmayan bir takımda kullanıcı hikayelerine değer atamak ve öncelik belirlemek büyük zorluk oluşturur.

İş birliği:
Şelale modelinde görev tanımları nettir ve görevler proje yöneticileri tarafından verilir Bu nedenle proje elemanlarının kendi işlerini yapmaları yeterlidir. İşini iyi yapan personel de bundan dolayı ödüllendirilebilir. Çevik yöntemlerde ise esas olan takımdır ve görev tanımları net değildir. Bu yüzden takımın görevini yerine getirebilmesi için takım elemanlarının çapraz fonksiyonlu olmaları gerekir. Başarılı ya da başarısız olan takımdır bu nedenle bireysel ödül veya ceza olmaz. Şelale modelinde proje süreçlerinde icra edilen görevler proje elemanlarına paylaştırılır ve süreç sonunda yapılan işler toparlanarak tamamlanır. Çevik yöntemlerde ise icra edilen faaliyetler takım bütünlüğü içinde icra edilir ve proje küçük parçalara ayrılarak bütün halinde çalışılır. Bu nedenle çevik yöntemlerde görev alan proje elemanlarının projenin bütünü ve genel hedefleri konusunda farkındalık seviyesi daha yüksektir.

Yerleşim:
Büyük şirketlerin dünyaya yayılmış olmalarının yanında farklı ülkelerdeki işgücünün daha ucuz olması sebebiyle proje bazlı işe alımlar da yapılmaktadır. Proje elemanlarının coğrafi olarak geniş bir alana dağıldığı durumlarda çevik yöntemlerin temel dayanaklarından iletişim zarar görür. Proje elemanlarının çalışma saatleri, büyük zaman dilimi farklılıkları nedeniyle eşzamanlı olamayabileceği için önemli toplantılara katılım konusunda problemler yaşanır, koordinasyonda güçlük çekilir. Telekomünikasyon vasıtalarıyla sağlanan iletişimde yakınlık hissi yüz yüze iletişimdeki kadar yoğun olmaz ve dayanışma azalır. Şelale modelinde yerleşim çevik yöntemler kadar önemli değildir çünkü projenin belli parçaları belli ekiplere bölünebilir fakat çevik yöntemler bu tarz bölünmelere izin vermez.

Ekip Çalışması:
Ekip çalışmasının kalitesini belirleyen başlıklar,

- iletişim,
- koordinasyon,
- üye katkısı dengesi,
- karşılıklı destek,
- çaba ve
- uyumdur.

Ekip çalışmasının şelale modeli ve çevik yöntemlerdeki kalitesinin karşılaştırılması için bu alt maddelerin karşılaştırılması gerekir. İletişim, şelale modelinde formeldir ve daha çok proje yöneticisine yazılı durum raporları şeklindedir. Çevik yöntemlerde ise enformel bir iletişim mevcuttur ve ayaküstü toplantılar, kısa sohbetler, ekran karşısında anlatım şeklinde gerçekleşir.

Koordinasyon şelale modelinde proje yöneticisinin kontrolü altındadır ve karar verme, tahminlerin yapılması, önceliklerin belirlenmesi ve görev dağılımı proje yöneticisi tarafından icra edilir. Çevik yöntemlerde bu uygulamaların hepsi otonom takımlar tarafından gerçekleştirilir.

Şelale modelinde üye katkısının dengeli olması söz konusu değildir. Proje yöneticisi kimin ne iş yapacağına karar verir ve proje elemanları kendilerine verilen görevleri yapmakla mükelleftirler. Çevik yöntemlerde çapraz fonksiyonlu takımlar içinde bütün takım elemanlarının katkısı beklenir ve bu husus günlük toplantılarda koordine edilir.

Karşılıklı destek açısından bakıldığında şelale modelinin hiyerarşik yapısı takım üyelerinin arasında karşılıklı bir desteği zorlaştırırken çevik yöntemlerde kodun bütün takıma ait olması, günlük toplantılar, retrospektif incelemeler karşılıklı desteği ve dayanışmayı canlandırır.

Şelale modelinde çaba, bireylerin kendilerine verilen görevleri tamamlamaları için harcanır. Çevik yöntemlerde ise çaba takıma aittir ve bireysel iş yoktur.

Şelale modelindeki hiyerarşik yapı ve formel iletişim, takım içi uyumu olumsuz etkilerken çevik yöntemlerde genellikle aynı fiziksel alanda bulunan takım elemanları arasındaki etkileşim ile daha yüksek bir uyum yakalanır. Bu sebeplerle ekip çalışması bakımından çevik yöntemler şelale modeline göre daha kaliteli bir yaklaşım ortaya koyar ve ekip çalışmasının önemli olduğu projeler için çevik yöntemler daha uygundur.

5.6. Üretilen değer (Product)

Nitelikler: Yazılımdan beklenen nitelikler, yazılımın uygulama alanına göre değişiklik gösterir. Örneğin bir banka sisteminden güvenli olması, iletişim ağlarının devamlılığının olması, interaktif bir oyunun hassas bir tepkisellik sunması beklenir. Ürün nitelikleri aşağıdaki şekilde genellenebilir:

- Sürdürülebilirlik*:* Yazılım müşterilerin değişen ihtiyaçlarına göre evrilebilecek şekilde üretilmelidir.
- Güvenilebilirlik ve Emniyet: Güvenilir yazılım, bir hata halinde fiziksel veya maddi hasar oluşturmamalıdır. Kötü niyetli kullanıcılar sisteme erişip zarar verememelidir.
- Verimlilik: Yazılım, işlemci ve RAM gibi sistem kaynaklarını boşa harcamamalıdır.
- Kabul Edilirlik: Yazılım, hitaben üretildiği kullanıcı tarafından kabul edilmelidir. Anlaşılır, kullanılabilir ve diğer sistemlerle uyumlu olmalıdır.

Bu niteliklere metodolojiler açısından bakılacak olursa, sürdürülebilirlik ve kabul edilirlik nitelikleri açısından çevik yöntemler şelale modeline göre daha uygundur. Şelale modeli değişimi kabul etmezken çevik yöntemlerde değişim hoş karşılanır. Çevik yöntemlerde yinelemeler sonunda yapılan ürün teslimleri ve alınan geri beslemeler ile müşterinin proje sürecinin tamamına katılımı, teslim sürecinde kabul edilirlik açısından şelale modeline göre avantaj sağlar. Güvenilebilirlik ve emniyet ile verimlilik nitelikleri açısından ise şelale modeli ön plana çıkmaktadır. Şelale modelinde uygulanan katı süreçler ile test yöntemleri, çevik yöntemlerin test ve kalite yöntemlerine göre daha güvenilirdir. Ayrıca ürünün sık ve çabuk teslim edilmesi ve müşteri gereksinimlerinin işlevselliğine odaklanılması, çevik yöntemlerde güvenlik, verimlilik gibi konuların göz ardı edilmesine sebep olabilir.

Kalite Güvence ve Hata Toleransı:
Çevik yöntemlerde eşli programlama, önce test yaklaşımı, gözden geçirme toplantıları, ürün değerlendirmeleri gibi tekniklerle sürekli inceleme ve ürünün geliştirme boyunca iyileştirilmesi sağlanmaya çalışılsa da, bu önlemlerin hata toleransı olmayan kritik sistemlerde istenen kalite için yeterli olmayacağı değerlendirilmektedir. Bu sistemlerde arzu edilen kalitenin sağlanabilmesi için icra edilen analiz, geliştirme ve test faaliyetleri daha detaylı, resmi ve katı uygulamalar içermelidir. Çevik yöntemlerin başlıca avantajları olan erken değer üretimi, sürekli geri besleme ve değişen şartlara adapte olabilme yeteneği, ilk ve en büyük önceliği hataya karşı sıfır tolerans olan bu sistemler için bir önem ifade etmez. Bu nedenle kalite güvence anlayışı açısından şelale modeli, çevik yöntemlere göre daha uygun bir yöntemdir.

Değer Üretimi:
Gerek şelale modeli gerekse çevik yöntemler yazılımın üreteceği değere önem verir. Kazanılmış değer analizi, yazılım projelerinin önemli bir yönetim alanıdır. Çevik yöntemlerde müşterinin istediği yazılım değeri proje süreci boyunca öğrenilir ve üretilir. Şelale modelinde

ise proje başlangıcında anlaşılması ve proje sonunda üretilmesi öngörülür . Şelale modelinde proje sonunda ürün teslimi yapıldığı için değer üretimi proje sonunda gerçekleşir. Çevik yöntemlerde ise yinelemeler sonunda teslim edilen her çalışan ürün müşteri için bir fayda sağlar. Yazılımdan beklenen değerin iyi anlaşılması için, müşterinin isteklerini iyi ifade edebilmesi gerekir. Şelale modelinden çevik yöntemlere geçen bir şirketin bir proje yöneticisi, gereksinimlerin son kullanıcı tarafından değil de kullanıcı olmayan bir müdür tarafından tanımlandığı için yaptıkları işlerin % 80"inin kullanıcı için bir değer üretmediğini gördüklerini belirtmiştir.

Kullanıcı Deneyimi:
Şelale modelinde kullanıcı deneyimi ve etkileşimi için detaylı bir tasarım ve geniş bir zaman ayrılır. Çevik yöntemlerde ise uzun vadeli plan ve detaylı tasarım yapılmaz ve kısa zamanlı yinelemeler, bütünsel bir yaklaşım gerektiren kullanıcı deneyimi ve etkileşimi için yapılacak çalışmalar açısından uygun değildir. Bunun yanında çevik yöntemlerde teslimatlar sonrası alınan geri beslemelerle kullanıcı deneyimi iyileştirilebilir.

Dokümantasyon ve Modelleme:
Üretilen yazılımın kullanılması öngörülen sürenin uzun ve kullanıcıların değişken olduğu durumlarda dokümantasyon bir başvuru kaynağı olarak önem arz edecektir. Geliştirme ekibinin dağınık olduğu veya bir altyüklenici kullanılan durumlarda da dokümantasyon önemli bir iletişim kanalı olarak kullanılabilir. Dokümantasyon ve modelleme, müşteri için bir değer üretmediği durumlarda bile geliştiriciler için faydalı olabilir. Evrilen büyük sistemlerin geliştirilmesinde başvurulacak mevcut modelleme yoksa, geliştiriciler sistemi anlamak ve yapılacak değişimin etkisini belirleyebilmek için çalışan kodu analiz etmek zorunda kalırlar. Dokümantasyon açısından uyguladıkları felsefe nedeniyle böyle durumlarda çevik yöntemlerden ziyade şelale modeli ile yazılım projesinin yönetilmesi daha faydalı olacaktır.

Teslimatlar:
Jenerik yazılım üreterek pazara sunan firmalar, sürümden belli bir süre önce ürünün pazarlaması için çalışmalara başlarlar. Şelale modelinde proje sonunda tek bir teslimat yapılması, bu felsefeye uygundur fakat çevik yöntemlerdeki kısa süreli yinelemeler sonucu verilen ürün versiyonları ve gereksinim önceliklerinin belirlenmesindeki esnek yaklaşım, pazarlama departmanlarına hangi özellikteki ürünü ne zaman pazarlayacakları konusunda net bir görüş sağlayamaz. Bu nedenle jenerik yazılım üreten firmalar için uzun vadeli planlamaları gereği şelale modeli daha uygun bir yaklaşımdır. Bunun yanında, pazar şartlarının ve iş süreçlerinin hızlı değiştiği, değişikliklere çabuk reaksiyon gösterilmesi gereken ortamlar ve sürekli değer üretimi beklenen projeler için, yinelemeler sonunda ürün tesliminin yapılması, çevik yöntemleri avantajlı kılar. Teslim edilen ürünlerden alınacak somut geri beslemeler sayesinde hatalar ayıklanır, düzeltmeler yapılır ve proje süreci daha sağlıklı ilerler.

Yeniden Kullanılabilirlik:
Çevik yöntemlerde basitlik prensibiyle hareket edildiği için, temel ihtiyacı karşılamak için çaba harcanır. Ürünün hiç gerçekleşmeyebilecek durumlar için genelleştirilmesine ve kapsamın genişletilmesine izin verilmez. Yeniden kullanılabilir bir ürün için bu şartların tanımlandığı dokümantasyon da gerekecektir ve bu da iş yükünü gereksiz yere arttıracaktır. Bu nedenle üretilen ürünler daha spesifiktir. Dokümantasyon ve modelleme ihtiyaç duyulmadığı sürece detaylı olarak üretilmediği için ürünlerin yeniden kullanılabilirliği çok zayıftır. Genel maksatlı, kullanımı genişletilebilecek ya da farklı projelerde yararlanılabilecek yazılım için, dokümantasyon ve modelleme imkânı nedeniyle şelale modeli daha uygundur.

Kullanım Onayı:
Üretilen yazılımın, kullanılacağı alan gereği bir otorite ya da düzenleyici kuruluş tarafından onaylanması gerekiyorsa (örneğin havacılık düzenlemeleri, adli düzenlemeler,) bu onaylama sırasında incelenecek dokümantasyonun detaylı olması gerekecektir ve söz konusu onay son ürüne verileceği için ara ürün teslimatı yapılmasının hiçbir önemi yoktur. Bu tarz yazılımlar için şelale modeli daha uygun bir yaklaşımdır.

5.7. Gereksinimler (Requirements)

Belirsizlik:

Şelale modeli, plan tabanlı ve önceden tanımlanan süreçler olduğu için belirsizlik toleransı yoktur. Çevik yöntemler ise planların değişeceği varsayımıyla önceden detaylı plan yapılmaması ve planlamadan ziyade ürün ortaya koyularak zamanın harcanması felsefesini benimsediği için belirsiz ortamlar açısından şelale modeline göre daha uygundur [41].

İhtiyaca Yabancılaşma:

Bir projedeki kullanıcı tarafından belirtilen ihtiyaçlar sırasıyla gereksinimlere, tasarım modeline, yazılım koduna ve son olarak ihtiyacı karşıladığı değerlendirilen ürüne dönüşmektedir. Şelale modelinde projenin başındaki gereksinim analizi asıl ihtiyaç ile etkileşim halinde iken müteakip aşamalarda bir önceki aşamanın çıktısı ile işlem yapılır. Projenin tamamlanma süreci göz önüne alındığında gereksinim analizinden kabul dönemine kadar ihtiyaca yabancılaşma söz konusu olur. Çevik yöntemlerde ise her yineleme başlangıcında gereksinimler gözden geçirilir. Müşteri ise her yineleme sonucu teslim edilen ürünün ihtiyacını karşılayıp karşılamadığını görür. Bu nedenle ihtiyaca yabancılaşma etkisi bir yineleme süresi ve boyutu ile sınırlı kalmaktadır.

Değişime Açıklık:

Çevik yöntemlerin ortaya çıkmasının en büyük sebeplerinden ve çevik yöntemlerin en büyük avantajlarından biri gereksinimlerdeki değişikliklerin proje boyunca kabul edilmesidir. Şelale modelinde gereksinim analizi süreçsı tamamlandıktan sonra projenin ilerleyen süreçlerinde gereksinimlerde yapılacak değişiklikler hem maliyeti arttıracağı hem de önceki süreçlera dönülmesini gerektireceği için kabul edilmez. Çevik yöntemlerde değişim maliyetleri şelale modeline göre nispeten daha düşüktür.

Uzun şartname hazırlıkları, bütçe-ödenek ve ihale süreçleri ile proje süreleri göz önüne alındığında bazen şelale modeli ile üretilen yazılımlar yıllar önceki ihtiyaçlara cevap vermek durumunda kalmaktadır.

Anlaşılırlık, Uygulanabilirlik, Güvenilirlik:

Geliştirme faaliyetine başlanmadan önce detaylı bir gereksinim tanımlama ve tasarım ihtiyacı var ise ayrılan zaman ve süreçler nedeniyle şelale modeli daha uygun bir yaklaşımdır. Fakat bu süreçlerin başarıyla aşılabilmesi için proje başlangıcında bütün ihtiyaçların biliniyor olması ve bu ihtiyaçların müşteri tarafından doğru ve uygulanabilir bir şekilde ifade edilebilmesi gerekir. Şelale modelinde, gereksinimler tam anlaşılmadığında müşteriyi temsilen kimse bulunmadığı için geliştiriciler iş süreçleri ve gereksinimler konusunda varsayımlarda bulunmak zorunda kalırlar. Geri dönüşler de kabul edilmediği için bu konulardaki aksaklıkların yaratacağı sıkıntılar proje boyunca hissedilecektir. Gereksinimlerin anlaşılırlığı, uygulanabilirliği ve güvenilirliği ile ilgili bir belirsizlik veya tereddüt varsa esnek yapısı nedeniyle çevik yöntemler daha uygun bir yaklaşım sunabilir

Gereksinim Yönetimi:
Şelale modelinde detaylı hazırlanan gereksinimlerin tasarım ile birleşerek kodlanacak görevlerin oluşturulmasıyla gerek proje yöneticileri gerekse dokümantasyon vasıtasıyla iyi bir gereksinim yönetimi sağlanmış olur. Çevik yöntemlerde ise gereksinimlerin müşteriler ile birlikte belirlenip önceliklendirilmesi ve geliştirme takımının bu sürece dahil olması küçük projelerde bir sorun yaratmasa da gereksinimlerin birden fazla grup tarafından tanımlandığı ve geliştirme takımlarının hepsiyle irtibatının olmadığı büyük projelerde gereksinim yönetimi eksikliği yaşanabilir.

5.8. Kaynaklar (Resources)

Şelale modelinde çoğu zaman tercih sebebi olan daha detaylı analiz, dokümantasyon ve testlerin yapılabilmesi için daha çok bütçe, zaman ve işgücü gerekir. Çevik yöntemler analiz, dokümantasyon ve test ihtiyaçlarının daha az olması sebebiyle kullanılan kaynaklar konusunda şelale modeline göre avantaj sağlar.

Zaman:
Şelale modelinde süreç bitimi için bütün işlerin bitmesi gerekir ve henüz bitirilmemiş işler yüzünden süreç ilerlemediğinde işini bitiren personel zaman kaybeder. Proje elemanlarının hep beraber takım olarak çalışması kaynakların daha etkin kullanımını sağlar [Bu nedenle şelale modelinde zaman israfı yaşanması mümkün iken çevik yöntemlerde kısa süreli yinelemeler sayesinde zaman en etkin şekilde kullanılır.

İş gücü:
Yazılım geliştirme sürecinde yapılan faaliyetlerin aynı olduğu, bu faaliyetlerin sıralamasında ve takibinde farklılıklar olduğu göz önünde bulundurularak, şelale modeli ile çevik yöntemlerin, adam-saat işgücü açısından karşılaştırılması için belirleyici iki faktör mevcuttur. Bunlar şelale modelinde detaylı bir şekilde yapılan analiz ve gereksinim tanımlamaları ile çevik yöntemlerde yapılmış olan işlerde ortaya çıkan değişiklikler yüzünden tekrar yapılan işlerdir. Çevik yöntemlerde gereksinimlerin tanımlanması konusunda tasarruf edilen zaman, değişiklikler yüzünden yeniden yapılan işlere harcanan zamandan fazla olduğu takdirde, adam-saat karşılaştırılması açısından çevik yöntemler daha avantajlıdır. Değişiklik nedeniyle yapılmış olan işlere tekrar harcanan işgücü gereksinim tanımlamasında tasarruf edilen işgücünden fazla ise şelale modeli daha avantajlıdır.

5.9. Risk Yönetimi (Risk Management)

Risk yönetimi, risk faktörlerinin tanımlanması ve her risk faktörünün gerçekleşme ihtimali ile potansiyel etkisinin analizi, risk faktörlerinin önceliklendirilmesi, gerçekleşme ihtimallerin düşürülebilmesi için risk azaltma stratejilerinin belirlenmesi ve probleme dönüşen risk faktörlerinin olumsuz etkilerinin en aza indirgenmesi faaliyetlerini kapsar. Söz konusu risk türleri, şelale modeli ve çevik yöntemler açısından değerlendirilecek olursa; zaman riskleri ve bütçe riskleri açısından şelale modelinde proje başlangıcında yapılan planlamaların detaylı ve gerçekçi bir şekilde yapılması önem arz etmektedir. Çevik yöntemlerde ise yinelemeli geliştirme anlayışı ve her yineleme sonunda yapılan teslimatlar bu risklerin izlenmesi ve yönetilmesini kolaylaştırmaktadır. Bunun yanında çevik yöntemlerin değişime açık yapısı, zaman ve bütçe risklerinin yönetimi açısından dezavantaj oluşturmaktadır. Şelale modeli ile çevik yöntemlerin yönetim anlayışındaki farklar nedeniyle yönetim risklerinin ele alınması da farklı olmalıdır. Şelale modelinde kontrol, otorite ve yönetim konuları, çevik yöntemlerde ise iletişim, organizasyon ve iş birliği konuları üzerinde durulmalıdır. Teknik riskler ve program risklerinin yönetimi açısından çevik yöntemlerin yineleme ve teslimatları izlenebilirliği kolaylaştırıp avantaj sağlarken, şelale modelinde kapsam konusunda değişikliğe kapalı olunması olası problemleri indirgemektedir. Çevik yöntemlerin personel bilgi ve tecrübesine dayalı yapısı nedeniyle personel risklerinin yönetimi konusu, şelale modeline göre daha fazla önem arz etmektedir.

Risk Türleri:

Zaman Riskleri:

Projenin, yanlış görev ve malzeme paylaşımından dolayı beklenen süre içerisinde gerçekleşmesine veya tamamlanmasına engel olan risk çeşididir.

Bütçe (Maliyet):

Gerçekçi olmayan bütçe tahminleri sonucu finansal sorunlara yol açan risklerdir. Bu risklerin gerçekleşmesi durumunda Tablolar değişmekte, maliyetler artmaktadır.

Yönetim Riskleri:

Amaçların net olmayışı, planlama eksikliği, yönetim tecrübesi ve eğitim eksikliği, iletişim sorunları, örgütsel sorunlar, otorite eksikliği ve kontrol problemlerini kapsamaktadır.

Teknik Riskler:

Genelde fonksiyonların yanlış olmasından kaynaklanır. Müşteri taleplerinin sürekli değişmesi, gelişmiş tekniklerin kullanılmaması ve geliştirilecek olan projenin zor faaliyetler içermesi gibi sebeplerden kaynaklanmaktadır.

Program Riskleri:

Proje kapsamının dışına çıkan, kontrol dışı durumlardan veya önceliklerin sürekli değişmesinden doğan risklerdir.

Sözleşme ve Yasal Riskler:
Değişen ihtiyaçları, pazar odaklı programları, sağlık ve güvenlik sorunları, hükümet düzenlemeleri ve ürün garantisi konularını içerir.

Personel Riskleri:
Personel duraklamaları, deneyim ve eğitim sorunları, etik ve ahlak konularını, personel çatışmalarını ve verimlilik sorunlarını içermekedir.

Diğer Kaynaklı Riskler:
Mevcut olmayan veya geç teslim edilen ekipman ve sarf malzemeleri, yetersiz aracı, yetersiz tesisleri, dağıtılan bölgeleri, bilgisayar kaynaklarının olmayışı ve yavaş tepki sürelerini kapsamaktadır.

5.10. Yazılım Güvenliği

Sahaya sürülen yazılımların güvenlik açısından incelendiği raporlar, başarılı görülen projelerin aslında birçok güvenlik sorunlarına ve programlama hatalarına sahip olduğu gerçeğini ortaya çıkarmaktadır.

Yazılımların karmaşıklığı ve genişleyebilirliği, ihtiyaçları karşılamak için gittikçe artmakta ve sonuç olarak hata olma olasılığı da oldukça yükselmektedir.

Yazılım güvenliğinde amaç, yazılımı geliştirirken sürdürülebilir işlevleri olacak şekilde inşa etmektir. Yazılım geliştirme yaşam döngüsü boyunca uygulanması öngörülen tasarımı, güvenli yapmak, güvenli kodlama ilkelerine uymak, yazılımın güvenli olduğundan emin olmak, tasarımı geliştirenleri güvenlik konusunda eğitmek gibi birçok uygulamayı kapsar.

Uygulama güvenliği, hataları daha çok, hatalar baş gösterdikten sonra bulup çözmeye dayanırken, yazılım güvenliği bu hataların olabileceğini başta tespit edip bu hatalara karşı dirençli bir yazılım geliştirmeyi hedefler.

NASA'ya göre minimum güvenlik güvencesi programı aşağıdaki faaliyetleri kapsamalıdır:

1. Güvenlik risk analizi
2. Geliştirilmekte ve/veya bakılmakta olan yazılım ve veri için güvenlik gereksinimleri
3. Geliştirme ve/veya bakım aşaması için güvenlik gereksinimleri
4. Tüm yazılımı gözden geçirme ve denetimlerinde güvenlik gereksinimlerinin değerlendirmesi
5. Konfigürasyon yönetimi ve düzeltici faaliyet süreçlerinin varolan yazılım için güvenliği sağlaması ve değişim değerlendirme süreçlerinin güvenlik ihlallerini önlemesi
6. Yazılım ve veri için fiziksel güvenliğin uygun olması
7. Varolan süreç modelleri ve standartlar

6. Yazılımın Test Edilmesi

Yazılımı bir sistem ve sinyaller organizasyonu olarak düşünmek gerekir. Girdi sinyallerine karşı bir çıktı sinyalleri üretebilir; girdisi olmadan çıktı üretebilir; geri besleme ve girdiye bağlı olarak çıktı üretebilir.

Geliştirilen yazılımlarda, kaynağı ve sebebi değişmekle birlikte çeşitli hataların olması kaçınılmazdır. Çalışılan uygulama alanının kritikliğine göre bu hataların doğuracağı sonuçların biçimi değişebilmektedir. Bir finans uygulamasında yapılan küçük bir hata çok büyük miktarlarda para kaybına yol açabilecekken, bir askeri uygulamada yapılması muhtemel küçük bir hata mal kaybının yanı sıra can kaybına da neden olma riskini taşımaktadır. Bu nedenle ***profesyonel kullanımı planlanan tüm yazılımların içerisindeki hataların bulunması ve düzeltilmesi gereklidir***.

Yazılım test çalışmaları, geliştirilen ya da geliştirilmekte olan yazılımlar içerisindeki hataların bulunup, düzeltilmesi, yazılımların hata içermediğinin garantilenmesi ve yazılımların doğru çalıştığının gösterimi amacı ile gerçekleştirilen faaliyetlerdir. Bu çalışmalar, geliştirme sürecinde gereksinimlerin belirlenmesi aşamasında başlayıp, yazılımların (Yazılım bir sistemi tanımalamaktadır.) müşteriye teslim aşamasına kadar devam eden bir süreçte belirli disiplinlere uygun olarak çalışılmasını gerektirmektedir.

Yazılım test faaliyetleri sadece geliştirilen kod parçaları üzerindeki hataların bulunması faaliyetlerini değil, geliştirme çalışmaları sırasında hata oluşmasını önleyecek yöntem ve yaklaşımların belirlenmesi faaliyetlerini de içerir. ***Geliştirme sürecinin başlangıç aşamalarında tespit edilen bir hatanın maliyeti ile müşteriye teslim edilmiş bir sistem üzerinde tespit edilen bir hatanın maliyeti arasında ciddi farklar olabilmektedir.*** Bu nedenle olası hataların geliştirme aşamasının mümkün olduğunca erken safhalarında bulunması için yapılan gözden geçirme ve tasarım doğrulama faaliyetleri gibi çalışmalar da doğrulama ve geçerlileme sürecinin bir parçasıdır.

Günümüz sistemlerinin tek bileşeni yazılım olmayan, yazılım dışında elektronik, mekanik ve elektromekaniksistemler, algılayıcılar, tartıcılar ve mukayes ediciler (filtreler) gibi bileşenleri de içerdiğinden, çok sayıda uzmanlık grubu bir arada çalışmaktadır. Bu nedenle yazılımların doğrulama süreçleri içerisindeki aşamalar, çıktılar, roller ve kullanılan araçlar ***hata tespit ve düzeltme faaliyetlerinin yanı sıra hata önleme amacına da dönük olmalıdır***.

Test, bir sistemi manuel veya otomatik yollarla deneyerek veya değerlendirerek, belirlenmiş gereksinimleri karşıladığının doğrulanması veya beklenen ile gözlenen sonuçlar arasındaki farkların belirlenmesi sürecidir.

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. Bu kapsamda dikkat edilmesi gereken hususlar şunlardır:

- ***Dinamik*** (Dinamik: değişken; statik: sabit) olarak yazılım mutlaka çalıştırılarak test edilmelidir.
- Yazılımın neredeyse sonsuz sayıda olabilecek çalışma alanlarının tümünün testi imkânsız olacağından; kritiklik düzeylerine göre sıralanıp, ***yeterli görülen sayıda, en kritikleri*** test edilmelidir.
- Test edilecek davranışın doğasına uygun ve ***muhtemel riskler***i göz önünde bulunduran testlerin gerçekleştirilmesidir.
- Test edilecek yazılımın, ***kullanıcı beklentilerine***, gereksinimlerine ve akla uygun, mantıklı beklentilere cevap verebildiğinin test edilmesidir.

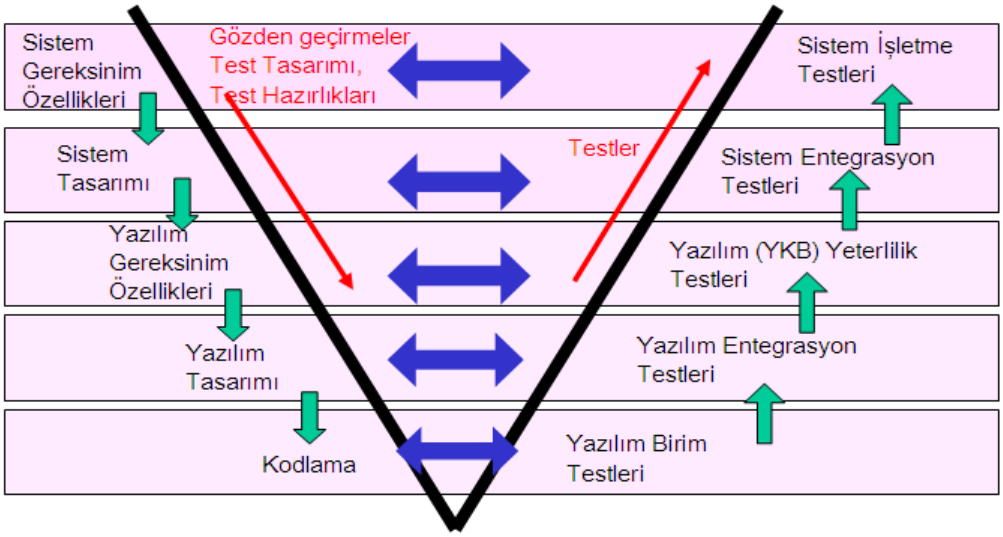
İşlevsel testler:

- uygunluk,
- tamlık,
- birlikte çalışma,
- hatalı girdi
- senaryo testlerini kapsar

Performans ve güvenilirlik testlerini kapsayan testler ise işlevsel olmayan testlerdir. Karşılaşılacak kullanıcı davranışına, veri hacmine uygun, gerçekçi bir şekilde ***performans testleri*** şekillendirilmelidir. ***Güvenilirlik testleri ise olgunluk, hata-toleransı, toparlanma ile ilgili testleri kapsamalıdır***.

Test faaliyetlerinin, yazılım geliştirme sürecinin daha başlangıç safhalarından itibaren vazgeçilmez bir parçası olduğu açıktır. Bu nedenle yazılımın tasarımı yapılırken, test planının da belirlenmiş olması, tasarımın test planına uygun özellikleri taşımasının sağlanması gerekmektedir.

Yazılım test faaliyetleri, tüm yazılım geliştirme süreci boyunca devam eden, sadece hataların bulunup ayıklanması işlemlerini değil, hata oluşmasını önleyici yaklaşımların uygulanmasını da içeren faaliyetlerdir. Test faaliyetleri aşağıda popüler bir gösterim şekli olan V-Modeli ile gösterilmiştir. V modelinde her tasarım aşamasına karşılık bir test aşaması mevcuttur.



Yazılım Geliştirme sürecinde test ile şu adımlar tanımlanır:

- Yayınlanan Sistem Tasarım Tanımı ve Yazılım Gereksinim Özellikleri incelenerek, Test konusundaki genel yaklaşım ve test faaliyetlerinin amacı göz önünde bulundurularak test aktivitelerinin planı, ***Yazılım Test Planı***, hazırlanır.
- Yazılım test planı çerçevesinde test platformlarının (test düzeneklerinin), kontrolünü ve veri akışı bilgilerini, yazılım ve donanım test araçlarını (***simülatörler, test sürücüleri***) ve veri hazırlıklarını içeren test alt yapısı gereksinimleri belirlenir. Test araçları sadece donanım olabildiği gibi donanım ve yazılımla birlikte de olabilir.
- Hazırlanan Yazılım Test Planı tanımlı sürece uygun olarak ***gözden geçirilir***. Yazılım mühendisinin kazanması gereken en önemli yeteneklerden biri gözden geçirme ve fark etmedir.
- **Testlerin planlanması** kapsamında yazılım **testlerinin hazırlık aşamas**ında, **sistem ortamın kurulmas**ında, **ön testler** sırasında, **testlerin gerçekleştirilme aşamas**ında **kimler**in sorumlu olacağı, testlerin gerçekleştirileceği ortamların planlanmasının nasıl yapılacağı, ***tanımlı süreç dışında*** projeye özel bir uygulama yapılıp yapılmayacağı, eğer farklılık varsa hangi aşamalarda ***süreçten farklılıklar*** olacağı, testler sonrasında **raporlama**nın nasıl ve kim tarafından gerçekleştirileceği, ***düzeltmeler***in ve sonrasındaki testlerin nasıl yapılacağı gibi sorulara cevaplar verilir.

Testler:

- ***İsteklerdeki eksiklikleri, hataları ve belirsizlik***leri belirlemek
- Kişisel hatalar: Benciller, psikopatlar, kendine aşırı güvenenler, yalnız kurtlar.
- Geliştirilen Algoritma ve Matematiksel Modellerde eksiklikleri, hataları ve belirsizlikleri belirlemek. İlk iki aşamada problem %80 oranında çözümler.
- Yazılım yazılırken hatayı bulmak
- Yazılım geliştirme aşamasında donanımın bileşenleri test etmek
- Kod yazılım sonrası bütünleştirme aşamasında sistem performansını test etmek
- Yazılım bütünleştirme ve uygulama aşamasında yapılan testler
- Kod çalışmaya başladığı andan itibaren gözden kaçan isteklerin belirlenmesi
- Bakım ve onarım aşamasında testler
- Gerçek zamanlı çalışırken yapılması gereken testler

Testle İlgili Gözlemler:

- "Test, hata bulma niyetiyle yürütülen bir program sürecidir." - Myers
- "Test, hataların varlığını gösterebilir, ancak onların yokluğunu asla gösteremez." - Dijkstra

İyi Test Uygulamaları:

- İyi bir test uygulaması ne içerir? Simülasyon (Benzeri-model), Optimizasyon (Sınırlama, sapma bulma),
- İyi bir test durumu, programın doğru şekilde çalıştığını göstermeyen durumlar için bir hatayı, tespit edilmemiş bir kusuru belirleme olasılığı yüksek olan bir işlevdir.
- Mümkünse kendi programınızı test etmeyiniz, kendi hatanızı bulmada zorlanırsınız. Kendiniz test etmek zorunda iseniz, kod yazmaya başladığınız andan itibaren adım adım sonuç takibi yapan test uygulamaları gerçekleyin.
- Her test aşamasına beklenen sonucun elde edilip edilmediğinin tespit edilmesi gerekmektedir.
- Yeniden üretilemez veya sinek testinden kaçının. Öylesine bir kod yazılım gerçeklenmiştir ki, geri dönüş olduğunda yazanlarda ne yaptıklarını bilemez durumdalar, hatanın nerede olduğunu bulmak imkansız. Çünkü sistematik değil ve planlı değil. Kod parçalarına açıklamının eklenmesi ve Readme (deneyim, fırsatlar ve hikaye içermeli) dokümanı oluşturulması gerekmektedir.
- Geçersiz ve geçerli giriş koşulları için test senaryoları yazınız.
- Her testin sonuçlarını iyice inceleyin
- Bir yazılım parçasındaki tespit edilen kusurların sayısı arttıkça, daha fazla tespit edilemeyen kusurların var olma olasılığı da artar.
- En iyi çalışanlarınızı test etmeye atayın
- Yazılım tasarımınızda test edilebilirliğin önemli bir amaç olduğundan emin olun.
- Programı daha kolay test etmek için asla değiştirmeyin
- Hemen hemen her diğer aktivite gibi testler hedeflerle başlamalıdır.

Yazılımlarda bellek sızıntısının ya da donanımsal hataların da olması durumu da sorun teşkil edebilecek önemli bir konudur. Birim testleri ile birlikte kod analiz araçları kullanılarak yazılımlar analiz edilmekte ve bellek sızıntıları tespit edilip düzeltilmektedir.

6.1. Yazılım Yeterlilik Testi

Yazılım yeterlilik testleri, kara kutu test (işlevlerin adım adım izlendiği durumların kaydı) yaklaşımıyla yazılımın, gereksinimlerini karşıladığının doğrulanması amacıyla yapılır.

Yazılım Yeterlilik testleri, Yazılım Gereksinim Özellikleri dokümanı temelinde şekillenir ve planlamanın ardından hazırlanmaya başlanır. Bu amaçla süreçte şu adımlar tanımlanmıştır:

- Yayınlanan Yazılım Gereksinim Özellikleri temel alınarak ve testlerin gereksinimleri yeterli düzeyde kapsamasına özen gösterilerek test tanımları belirlenir.
- Testin başlatılması, yerine getirilmesi ve gerekli verilerin toplanması için gerekli olan adımların sıralamasını, her bir adım için beklenen sonuçları ve değerlendirme kriterlerini içeren test yönergeleri hazırlanır.
- Test tanımlarını ve yönergelerini içeren **Yazılım Test Tanımları** dokümanı tanımlı sürece uygun olarak gözden geçirilir. Bu gözden geçirme sürecinde kontrol listesi referans olarak kullanılabilir.

Yazılım yeterlilik test tanımlarının hazırlanması için öncelikle yazılım ***gereksinimlerinin olgunlaşmış olması*** gerekmektedir. ***Gereksinimlerin tamamen veya kısmen belirsiz olması***, test sürecinin sağlam temeller üzerine oturmasını engeller. Bir gereksinimin değişmez hale getirilip dondurulmasının pratikte zor olduğu düşünüldüğünde uygulama için önerilebilecek çalışma yöntemi, yazılım gereksinimlerinin yazılımı geliştirenlerin deneyimleri doğrultusunda test edilebilecek olgunluğa erişmiş olduğu kanaatine varılması ile gereksinim dokümanının dondurulması, ve bir kopyasının alınıp, gereksinimlerin test edilebilecek düzeye erişip erişmediğinin kontrolü için test tasarım ekibinin görüşüne sunulmasıdır. Daha önceden aynı özelliklere sahip yazılımların testlerini muhtemelen gerçekleştirmiş olan test tasarım ekibi her bir gereğin nasıl test edileceğini ve test sırasında nasıl bir ortamın sağlanması gerektiği konusunda bütün gereksinim dokümanını gözden geçirmelidir. Test tasarım ekibi, gereksinimlerin test edilebilir olgunluğa erişmiş olduğunu onayladığında, dondurulmuş gereksinimlere göre test hazırlıklarına paralelden başlayabilir ve test hazırlıkları devam ederken bir yandan da gereksinimlerin olgunlaşması devam edebilir. Ancak test hazırlıklarının son aşamasında test tanımları ve gereksinimler arasında bir uyumsuzluk oluşup oluşmadığı kontrol edilmelidir.

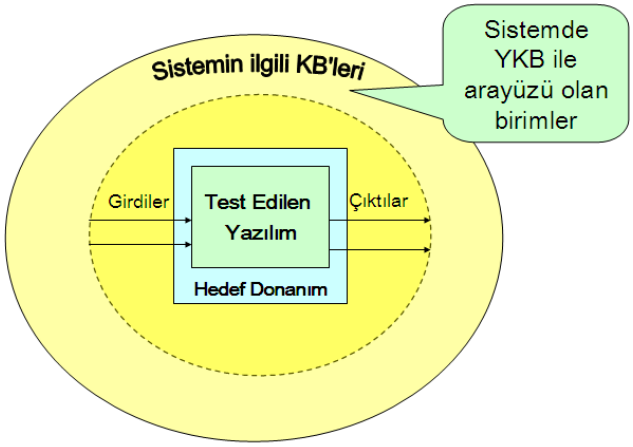
Yazılım gereksinimlerinin olgunlaşması ve test edilebilirliğe göre gözden geçirilmesi test tanımları hazırlanması aşamasına geçiş için olmazsa olmaz iki önkoşuldur: Test tanımları, doğruladığı gereksinimlerin yazılım tarafından karşılanıp karşılanmadığını bulmaya yönelik olmalı ayrıca olağan dışı girdileri de kapsayacak şekilde yazılmalıdır.

Test tanımlarının yazılması sırasında dikkat edilmesi gereken bir diğer konu da her bir gereksinimin en az bir test tanımı ile doğrulanmasının gerekliliğidir. Gereksinim Yönetim araçları kullanılarak gereksinimler ve test tanımları arasında izlenebilirlik kurulup test edilmemiş herhangi bir gereksinimin kalmaması sağlanmaktadır.

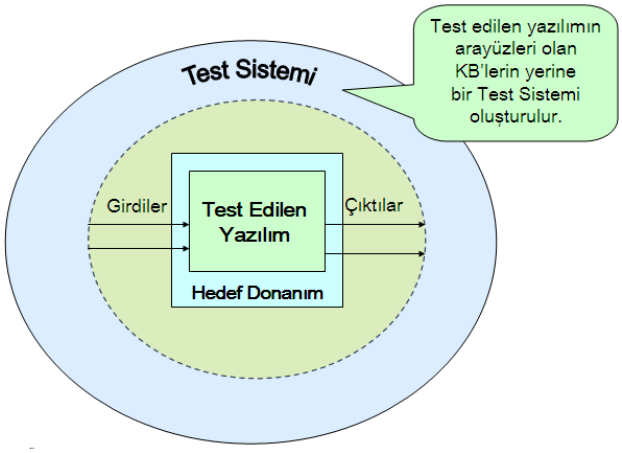
Hazırlanan Yazılım Test Tanımları dokümanında; testin amacı, testin hangi gerekleri adresleyerek doğruladığı, testin hangi konfigürasyonda gerçekleştirileceği, test yönergesi, testin başarılı sayılması için gerekli olan kriterler ve varsa test girdileri, önkoşulları, sınırlamaları ile çıktıları tanımlanmaktadır. Test yönergelerinin kapsamı ise doğrulanan gereğin kritikliği ve hata çıkma olasılığıyla doğru orantılı olmalıdır. Test sırasında çıkacak olası bir hatanın göreve etkisi ve sonuçları değerlendirilmelidir. Ayrıca testlerin belirlenmesinde; karmaşık, zaman içinde değişikliğe uğrayan, uygulama içinde yenilik olan (yöntem, yeni teknoloji vb.), optimizasyon gerektiren ve daha önce hata bulunan kısımlara özel önem verilmelidir.

6.2. Test Yazılımları Geliştirme Faaliyetleri

Yazılım yeterlilik testleri için testlerin kara kutu (İzleme, takip, kayıt) yaklaşımı ile gerçekleştirilebilmesi, yazılımın sistemden bağımsız bir test ortamı içerisinde koşması için önem taşımaktadır. Yazılımda oluşabilecek hataları, sistemde oluşabilecek hatalardan bağımsız olarak ayıklamak test işlemini kolaylaştırıp, verimliliği artırarak, hatanın olabildiğince erken aşamalarda yakalanmasını sağlar. Bu durum test edilecek yazılımın beraber çalıştığı her türlü yazılımın ve donanımın gerçek sistem üstünde çalışıyormuşçasına simüle edilmesi ihtiyacını doğurmaktadır. Bunun için Şekil -A'de görülen test edilen yazılımın ilişki içinde olduğu konfigürasyon birimlerinin (KB) yerine Şekil-B'deki gibi test sistemi yerleştirilmektedir. Test sistemi test yazılımlarından oluşmaktadır. Test yazılımları test edilecek yazılım ile simülasyonunu yaptığı yazılım veya donanımın ilgili arayüzünü kontrollü bir şekilde gerçekleyen yazılımlardır. Yani bir yazılımın testi sırasında o yazılımın haberleştiği yazılım veya donanımın simülasyonunun yapılması, o yazılım veya donanımın tüm işlevlerinin gerçekleştirilmesi değil sadece test için gerekli olan arayüzün kodlanmasıdır. Öte yandan test yazılımları, ilgili arayüzleri gerçeklerken, gerçek sistemde çalışan yazılımlardan farklı olarak o arayüzdeki olağan dışı durumları da gerçekleme olanağına sahip olmalıdır. Bir yazılımın tam anlamıyla test edilmesi için zaman zaman birden fazla ve farklı yazılım geliştirme ortamlarında kodlanmış test yazılımlarına ihtiyaç duyulabilmektedir.



Şekil-A: Yazılım Çalışma Ortamı



Şekil-B: Yazılım Test Ortamı

Gereksinimlerde yapılacak ufak bir değişiklik test yazılımlarında kökten değişikliklere yol açabilmektedir. Örneğin, test edilen yazılımın gereksinimlerinde olabilecek bir değişiklik test yazılımının tamamen başka bir ortamda tekrar geliştirilmesi gerekliliğini ortaya çıkarabilmektedir. Bu da işgücü ve zaman kaybına yol açabilmektedir. Bu yüzden testlerin planlama aşamasından sonra tasarım ekipleri tarafından gözden geçirilmesi önem kazanmaktadır.

Test yazılımlarının hazırlanması aşamasının en önemli girdileri yazılım test tanım dokümanı ve yazılım arayüz tasarım tanımı dokümanlarıdır. Her iki dokümanın da olgunlaşması test yazılımlarının geliştirilmesinin başlaması için gereklidir.

Otomatik test yazılımlarının kullanılması ile testlerin daha hızlı gerçekleştirilmesi sağlanabilmektedir. Bu amaçla kaydet-oynat yeteneğine sahip, kullanıcının, kullanıcı arayüzünde yaptığı davranışları kaydederek daha sonra sanki kullanıcı yapıyormuş gibi aynı hareketleri farklı girdilerle defalarca tekrarlayabilen araçlar kullanılabilmektedir. Ancak farklı

arayüzler üzerinden farklı ortamlara sahip çevre birimlerle etkileşimi çok olan gömülü yazılımların testi için bu gibi otomatik test araçlarının yetenekleri kısıtlı kalmaktadır. Bu gibi durumlarda pratik olarak önerilen yöntem, geliştirilmiş olan test yazılımı içerisine bu otomasyonun gömülmesi ve test yazılımı aracılığı ile testlerin mümkün olduğunca otomatik olarak gerçekleştirilmesinin sağlanmasıdır.

6.3. Testlerin Gerçekleştirilmesi

Kodun hazır olduğunun duyurulmasını takiben Yeterlilik testleri gerçekleştirilmeye başlanır. Bu amaçla süreçte şu adımlar tanımlanmıştır:

- Test tanımları ve test araçları olgunlaştırılır.
- Yazılım gereksinimlerinden test tanımlarına izlenebilirlik kontrol edilir.
- Üzerinde koşacağı Donanım Konfigürasyon Birimi hedef prototipi ve test yazılım ve araçlarının bir araya getirilmesiyle bir test düzeneği kurulur.
- Test tanımları çerçevesinde yeterlilik testleri gerçekleştirilir. Hata oluşması durumunda problemlerin çözülmesi amacıyla ilgili sürece geçilir ve süreç işletilir.

Testler gerçekleştirilmeye başlanmadan önce ortamın (test konfigürasyonunun) testlere hazır olup olmadığının kontrolünün yapılması gereklidir. Hazırlanan test yazılımlarının testler öncesinde doğrulanmış olması önem taşımaktadır. Resmi testler öncesi test edilecek yazılımın test ortamına entegrasyonun gerçekleştirildiği ve yazılımın testlere hazır olup olmadığının kontrolü duman (smoke) testleri aracılığıyla yapılır. Bu testler, test edilecek yazılımın en temel yeteneklerini doğrulayan test tanımları arasından seçilmelidir. Ancak bu testler başarılı olarak gerçekleştirilirse, resmi olarak testlere başlanmalıdır.

Testlerin hangi sırayla yapılacağı doğrulanan testlerin kritikliği ve testlerin fonksiyonel olarak birbirleri ile olan bağlantılarıyla alakalı olabilir. Bu sıranın ne olacağına test planlama aşamasında karar verilmeli ve onaylanmalıdır.

Yazılıma yapılan ekleme veya düzeltmeler yeni hatalara sebep olabilmektedir. Bu hataların tespit edilmesi ve olası gerilemenin belirlenmesi amacıyla yapılan testler regresyon (gerileme) testleridir. Herhangi bir değişiklik sonucunda, yeni sürümü yapılan yazılımın ***regresyon testleri***nde, sadece yeni sürümdeki değişiklik bilgisinin incelemesi sonucunda gerekli olduğunu belirlenen testler yapılabilir.

6.4. Test Seviyeleri

1) Birim Testi
2) Entegrasyon - Bütünleştirme Testi
3) Doğrulama Testi
    a. Gerileme testi
    b. Alfa Testi
    c. Beta testi
4) Kabul testleri

Birim Testi

- Algoritmalar ve mantık
- Veri yapıları (global ve yerel)
- Arayüzler
- Bağımsız yollar
- Sınır şartları
- Hata işleme

Birim Testi, Birim Entegrasyon Testi

Geleneksel yaklaşımda daha çok görsel denetimle ya da manuel yöntemlerle yapılan birim testleri, otomatik test araçlarının kullanımıyla daha sağlıklı ve tekrarlanabilir olarak yapılmaya başlanmıştır. Test planının ve test tanımlarının hazırlanmasına paralel olarak, yazılımların kodlanmaya başlanması ile birlikte birim testleri gerçekleştirilmeye başlanmaktadır. Yazılım Geliştirme Sürecinde Birim Testlerine yönelik olarak süreçte şu adımlar tanımlanmıştır:

- Yazılım Birim test hazırlıkları yapılır. Birim, modül ve bunların çeşitli bileşenlerinden oluşan kümelerin hangi sırayla test edileceği, test gereksinimleri (donanım ve yazılım), durumları ve yöntemleri belirlenir.
- Hazırlık kapsamında eğer gerekiyor ise sürücü/kart entegrasyonu ve bunların testi gerçekleştirilir. Kart seviyesinde koşan yazılımların ve geliştirilen veya hazır alınan kartın spesifikasyonlarına uygunluğu doğrulanır.
- Üretilen kod için birim/modül/küme testleri uygun debug/test araçlarıyla gerçekleştirilir. Testlerin kayıtları tutulur.
- Yazılım Konfigürasyon Birimi yeterlilik testi için yazılımın hazır olduğuna karar verilmesi durumunda kodun hazır olduğu duyurulur.

Birim testleri bu süreç dahilinde manuel ya da test araçları kullanılarak otomatik olarak yapılmaktadır. Test araçları kullanılarak otomatik test yapılması hedeflendiğinde, kullanılacak olan aracın kodlama aşamasına geçilmeden önce belirlenmiş olması ve aracın özelliklerine uygun şekilde yazılımın gerçekleştiriminin yapılması gerekmektedir. Otomatik test araçları yazılım birimleri için yazılan birim testlerinin işaretlenmesi ve bu testlerin

dışarıdan erişilerek çalıştırılması mantığı ile çalışmaktadır. Bu araçlar test yordamlarının işaretlenmesini sağlayan işlevleri ve doğrulama cümlelerinin yazılabilmesini sağlayan (karşılaştırma işlemleri vs.) yordamları içeren kütüphaneyi ve testlerin gerçekleştirimini ve raporlanmasını sağlayan kullanıcı arayüzünü içermektedir.

Birim testi için manuel yöntemler kullanılması durumunda yazılım geliştirme ortamının ***hata ayıklama (debug) yetenekleri***nden faydalanılmaktadır. Performansa yönelik testler için ise koda eklemeler yapılarak zaman bilgisi tutulmakta, yazılım işlevini tamamladıktan sonra bu bilgiler diske kaydedilerek değerlendirilmesi manuel olarak yapılmaktadır.

Sistem Entegrasyon Testi

Yazılım yeterlilik testlerinin gerçekleştirilmesi sonrasında sistem tarafında da çalışmalar Sistem Entegrasyon Testi Hazırlıklarını Yap aşaması ile devam eder. Bu amaçla süreçte şu adımlar tanımlanmıştır:

- Her bir entegrasyon aşamasındaki gereksinimlere uygun olarak test tanımları Test mühendisleri tarafından belirlenir ve gerekli test araçları oluşturulur. Tasarlanması gereken özel amaçlı donanım ve yazılım test araçları ve yeni test yatırımları için Donanım Geliştirme Sürecine uygun olarak planlama yapılır.
- Test tanımlarının dokümante edildiği dokümanlar tanımlı sürece uygun olarak gözden geçirilir.
- Temin edilen birimler bütünleşik sistem elde edilene kadar ihtiyaca göre aşamalı olarak entegre ve test edilir.

Test hazırlıklarının tamamlanması sonrasında Sistem/Alt sistem Testini Yap ve Entegrasyon Test Sonuçlarını Gözden Geçir aşaması gerçekleştirilir. Bu amaçla süreçte şu adımlar tanımlanmıştır:

- Entegre edilen her test edilecek yapı üzerinde hazırlanan dokümanlara uygun doğrulama testleri uygulanır ve test sonuçları kaydedilir. Bu kayıtların analizi ve yorumlanması ile test raporu hazırlanır.
- Hazırlanan rapor tanımlı sürece uygun olarak gözden geçirilir.
- Sistem tasarımı sürecinde belirlenen ve Sistem / Alt Sistem Tasarım Tanımı (Sistem Tasarım Tanımı) dokümanında temel özellikleri oluşturulan DONANIM KONFIGÜRASYON BIRIMI ve YAZILIM KONFIGÜRASYON BIRIMI'ler geliştirildikten veya hazır olarak temin edildikten sonra DONANIM KONFIGÜRASYON BIRIMI ve YAZILIM KONFIGÜRASYON BIRIMI'lerin entegrasyonu aşamasına geçilir. Entegrasyonun doğrulanması amacıyla entegrasyon testleri gerçekleştirilir.

Entegrasyon testleri kapsamında öncelikle donanım ve yazılım arayüzlerinin doğruluğu kontrol edilir. Arayüzler doğrulandıktan sonra SISTEM TASARIM TANIMI dokümanında belirtilen sistem çalışma senaryoları kapsamında fonksiyonel ve performans testleri yapılır.
Test sonuçları Sistem / Altsistem Test Raporu (STER) ile raporlanır. Test sonuçlarına göre DONANIM KONFIGÜRASYON BIRIMI ve YAZILIM KONFIGÜRASYON BIRIMI'ler için gerekli düzeltmeler yapılır.

Neden entegrasyon testi gereklidir?

- Bir modül diğerini olumsuz etkileyebilir.
- Bütünleştirme yapıldığında, alt fonksiyonlar istenen ana fonksiyonu üretemeyebilir
- Bireysel olarak kabul edilebilir hesaplamardaki uygunsuzluk kabul edilemez seviyelere yükseltilebilir.
- Birim testinde tespit edilmeyen arayüz hataları görünebilir
- Zamanlama problemleri (gerçek zamanlı sistemlerde) birim testiyle tespit edilemez
- Kaynak uyumsuzluk sorunları birim testi ile tespit edilemez

Yukarıdan Aşağıya Entegrasyon:

1. Ana kontrol modülü bir sürücü olarak kullanılır ve ana modüle doğrudan bağlı tüm modüller için taslaklar kullanılır.
2. Seçilen entegrasyon yaklaşımına bağlı olarak (önce derinlik veya genişlik), alt diziler birer birer birer birer değiştirilir.
3. Her bir modül entegre edildiğinden testler çalıştırılır.
4. Bir dizi testin başarılı bir şekilde tamamlanması üzerine, başka bir birim modülle değiştirilir.
5. Yeni modüllerin entegrasyonu sonucunda hataların oluşmadığından emin olmak için regresyon testi yapılır.

- Developing stubs that can pass data up is almost as much work as developing the actual module

Yukarıdan aşağıya entegrasyon ile ilgili sorunlar

- Çoğu zaman, hesaplamalar hiyerarşinin altındaki modüllerde yapılır.
- Alt modüller tipik olarak daha yüksek modüle veri iletmez
- Alt seviye modüllere kadar geçikme testleri bir andan ziyade aynı anda pek çok modülün bütünleşik sonuçları hazırlanır.
- Verileri aktarabilen taslakların geliştirilmesi, neredeyse gerçek modülün geliştirilmesi kadar bir iştir.

Alttan Entegrasyon

- Entegrasyon, belirli bir yazılım alt işleyişini gerçekleştiren kümeler halinde birleştirilen ya da inşa edilen en alt düzey modüller ile başlar.
- Giriş ve çıkış test durumlarını koordine etmek için Sürücü yazılımları (kılavuzlar olarak geliştirilen kontrol programları) geliştirilir.
- Küme sınanmıştır
- Sürücüler kaldırılır ve kümeler, program yapısında yukarı doğru birleştirilir

Alt Yukarı Entegrasyon Problemleri

- Son modül bütünleşene kadar tüm program mevcut değildir.
- Süreç ve kaynak çekişme sorunları süreçte geç kalana kadar bulunmaz.

Doğrulama Testi

- Yazılımın YAZILIM İSTEKLERI TANIMLAMA'de tanımlanan tüm gereksinimleri karşılayıp karşılamadığını belirler
- Yazılı gereklilikleri şarttır
- Yazılımın, yazılımdaki değişiklikler ve değişiklikler ışığında tüm gereksinimlerini karşılayıp karşılamadığını belirlemek için regresyon testi yapılır.
- Regresyon testi, yeni testler geliştirilmeden, varolan validasyon testlerini seçici olarak tekrarlamayı içerir.

Alfa ve Beta Testi:

- Müşterilere, odaklanmalarını istediğiniz şeylerin taslağını ve yürütmeleri için belirli test senaryoları sunmak en iyisidir.
- Keşfettikleri kusurları düzeltme taahhüdüyle aktif olarak ilgilenen müşterilere sağlar.

6.5. Sistem Kabul Testi ve Arazi Testleri

Tasarım ve doğrulama süreci tamamlanan sistemin müşteri isteklerini karşılayıp karşılamadığı kabul testleri ile ispat edilir. Müşteri tarafından belirlenmiş bütün gereklerin doğrulanması için Muayene Kabul Dokümanı oluşturulur.

Sistem bu dokümana göre önce şirket içerisinde doğrulandıktan sonra müşteri davet edilir ve testler müşteri ile birlikte tekrarlanır.

Sistem kabul testlerinin, sistem özelliklerine bağlı olarak şirket içerisinde ve/veya arazide yapılması gerekebilir. Örneğin bir silah sisteminde sistemin en önemli özelliği olan atış yeteneğinin gerçek atışlar yapılarak doğrulanması gereklidir. Bu gibi durumlarda testler için uygun atış alanlarının kullanılması gerekmektedir.

Kabul testleri

• Müşterilerin mevcut veya doğrudan dahil olması haricinde, doğrulama testine benzer.
• Genellikle testler müşteri tarafından geliştirilir

Sistem Yeterlilik Testi

Sistem entegrasyon ve testleri tamamlandığında Sistem İşletme Testlerinin yapılması ile sistemin doğrulanması sağlanır. Bu amaçla süreçte şu adımlar tanımlanmıştır:

- Sistem İşletme Test Planı içinde yer alan test gereksinimlerinin sağlanmasında kullanılacak olan test tanımları yazılır.
- Sistem, gerçek kullanım ortamında veya benzer koşullarda hazırlanan test tanımlarına uygun olarak test edilir ve rapor oluşturulur.
- Raporlar Gözden Geçirme Sürecine göre gözden geçirilir ve doğrulanmış ve geçerli kılınmış sistem elde edilir.

Sistem yeterlilik testlerine yönelik çalışmalar Sistem Gereksinim Özellikleri dokümanı hazırlandıktan ve gözden geçirme süreci tamamlandıktan sonra başlar.
Sistem İşletme Test Tanımı dokümanına her bir gereksinim için doğrulama metoduna uygun olarak test tanımları yazılır ve testler uygulanır.

Yazılım tasarımı yoğun olan sistemlerde Donanım Konfigürasyon Birimi'lerin sistemde doğrulanması entegrasyon testleri sonrasında tamamlanmasına rağmen yazılım ile ilgili gereksinimlerin doğrulanması sistem işletme testleri sonunda tamamlanmaktadır.

6.6. Hata Önleyici Faaliyetler

Doğrulama faaliyetlerinin önemli bir parçası hatayı oluşmadan önlemeye yönelik çalışmaların yürütülmesidir. Bu çalışmalar, ileride ortaya çıkabilecek hataları önleyerek, test ve düzeltme maliyetlerini azaltmayı hedeflemektedir. Bu kapsamda yapılan başlıca faaliyetler gözden geçirme ve tasarım doğrulaması faaliyetleridir.

Gözden geçirme, sürecin her safhasında farklı ürünler için uygulanabilen bir yöntemdir. Yazılım geliştirme sürecinde hazırlanan dokümanlar, yazılım mimari tasarımları, detaylı tasarımlar ve yazılan kodlar için gözden geçirme gerçekleştirilmektedir.

Gözden geçirme çalışmaları için süreçte tanımlanan temel adımlar şunlardır:

- Ürünün gözden geçirmeye çıkarılması: Gözden geçirmeye hazır olduğu düşünülen ürün üzerinde değişiklikler dondurularak, gözden geçirme için belirlenen ekibe duyurulur. Belirlenen süre içerisinde ürünün incelenmesi ve gözden geçirme kaydının doldurulması istenir. Bu kayıtlar doldurulurken ürün için hazırlanmış olan kontrol listeleri de referans alınır.
- Gözden geçirme kayıtlarının incelenmesi: Belirtilen süre sonunda tüm gözden geçirme kayıtları ürünü hazırlayan kişi tarafından toparlanır ve kayıtların görüşülmesi için toplantı düzenlenir. Toplantı sırasında kayıtlar üzerinden geçilerek ürüne yansıtılacak değişiklikler belirlenir.
- Gözden geçirme kayıtlarının işlenmesi: Gözden geçirme toplantısı sırasında alınan kararlar doğrultusunda üründe değişiklikler yapılır.

Bu çalışmalar, ürünlere farklı paydaşların gözüyle bakarak erken safhalarda hataya sebep olabilecek hususların belirlenmesini ve düzeltilmesini sağlar.

Hatanın önlenmesine yönelik diğer faaliyetler tasarım doğrulama faaliyetleridir. Bu faaliyetler kapsamında, tasarım yapıldıktan sonra sistem senaryoları ele alınarak tasarımın tüm senaryoları karşılayıp karşılamadığı değerlendirilmektedir. Kullanıcı arayüzü tasarımlarının doğrulanması için ise hızlı prototipleme çalışması yapılarak ekranlar oluşturulmaktadır. Bu ekranlar üzerinde, yine senaryolar çalıştırılarak, işlevlerin akış sırasının ve ekranların amacına uygunluğu gözden geçirilir.

Bu yüzden testlerin planlama aşamasından, hazırlanma, gerçekleştirme ve raporlanma aşamalarına kadar geçen sürede uzman kişiler rol almakta, basit ve anlaşılır iş talimatları ile test süreci desteklenmektedir. Testler yapılırken mümkün olduğunca hazır veya projeye özgü geliştirilen test yazılımlarından destek alınmaktadır. Hatalar, hata takip araçlarından takip edilmektedir. Test tanımları gereksinim yönetim araçlarına girilerek gereksinimlerle izlenebilirlikleri sağlanıp test edilmemiş gereksinimlerin kalmaması garanti edilmektedir. Tüm bu doğrulama faaliyetleri sistemlerin güvenilir bir şekilde müşteriye ulaşmasını sağlamaktadır.

6.7. Test Metotları

- Beyaz kutu veya cam kutu testi
- Kara kutu testi
- Artımlı entegrasyon gerçekleştirmek için yukarıdan aşağıya
- Bir müşteri gibi davranma

6.8. Test Tipleri

- Fonksiyonel testler
- Algoritmik testler
- Pozitif testler
- Olumsuz testler
- Kullanılabilirlik testleri
- Sınır testleri
- Başlangıç / Kapama testleri
- Platform testleri
- Yük / stres testleri

Eşzamanlı Geliştirme / Doğrulama Test Modeli

- Gelişme devam ederken gayri resmi doğrulama yapmak
- Yazılım geliştirme sürecinde, doğrulama testlerinin geliştirilmesi ve hata ayıklanması için bir fırsat sunar.
- Yazılım mühendislerine erken geri bildirim sağlar
- Resmi validasyondaki sonuçlar daha az olağandır, çünkü problemlerin çoğu zaten tespit edilmiş ve tespit edilmiştir.

Doğrulama Hazırlık Değerlendirmesi

- Resmi olmayan doğrulama sırasında geliştiriciler, Yazılım İstekleri Tanımlama'ya uymak için gerekli değişiklikleri yapabilirler.
- Resmi olmayan doğrulama sırasında testleri Yazılım İstekleri Tanımlama'ya uygun hale getirmek için testleri çalıştırır ve gerekli değişiklikleri yapar.
- Resmi doğrulama sırasında yapılabilecek tek değişiklik, resmi doğrulama testi sırasında bildirilen hatalara yanıt olarak hata düzeltmeleridir. Şu anda yeni özellikler eklenemez.
- Resmi doğrulama sırasında, informal doğrulama sırasında aynı test seti tekrar çalıştırılır. Yeni test eklenmez.

Resmi Doğrulama Testi için Giriş Kriterleri

• Yazılım geliştirme tamamlandı (“tamamlandı”) nın kesin tanımı gereklidir.
• Test planı gözden geçirildi, onaylandı ve belge kontrolü altında.
• Yazılım İstekleri Tanımlama'da bir gereksinim denetimi gerçekleştirilmiştir.
• Yazılım Tasarım Tanımları'lerde tasarım denetimleri gerçekleştirilmiştir (Yazılım Tasarım Tanımları).
• Tüm “kritik modüller” üzerinde kod denetimleri gerçekleştirilmiştir.
• Tüm test komutları tamamlandı ve yazılım doğrulama testi prosedürü dokümanı gözden geçirildi, onaylandı ve belge kontrolü altına alındı.
• Seçilen test komutları gözden geçirildi, onaylandı ve belge kontrolü altına alındı.
• Tüm test senaryoları en az bir defa gerçekleştirilmiştir.
• CM araçları yerinde ve tüm kaynak kodu yapılandırma kontrolü altında.
• Yazılım problemi raporlama prosedürleri yürürlüktedir.
• Doğrulama testi tamamlama kriterleri geliştirilmiş, gözden geçirilmiş ve onaylanmıştır.

Resmi Doğrulama

- Resmi olmayan doğrulama sırasında yürütülen aynı testler tekrar yürütülür ve sonuçlar kaydedilir.
- Yazılım Sorunu Raporları (Yazilim Problem Raporlama'lar) başarısız olan her bir test için gönderilir.
- Yazilim Problem Raporlama izleme gerçekleştirilir ve tüm Yazılım Problem Raporlama'ların durumunu içerir (ör. Açık, sabit, doğrulanmış, ertelenmiş, bir hata değil)
- Düzeltilen her bir hata için, Yazilim Problem Raporlama, hatayı düzeltmek için değiştirilen modülleri tanımlar.
- Temel değişiklik değerlendirmesi sadece değişmiş olması gereken modülleri ve yeni özelliklerin girmediğinden emin olmak için kullanılır.
- Yeni hataların getirilmediğinden emin olmak için, informal kod incelemeleri değiştirilen modüller üzerinde seçici olarak yürütülmektedir.
- Hata bulup düzeltmek için gereken süre (find-fix cycle time) takip edilir.

Regresyon testi aşağıdaki yönergeler kullanılarak gerçekleştirilir:

- Hangi modüllerin ek testlere ihtiyaç duyabileceğini belirlemek için karmaşıklık önlemlerini kullanın
- Hangi testlerin tekrar çalıştırılacağına karar vermek için karar verin
- Yazılım tasarımı ve geçmiş tarih bilgisine dair temel karar
- Test durumunu izle (örn. Geçti, başarısız oldu veya çalışmadı).
- Yazılım güvenilirliği büyüme izlemesi için kümülatif test süresini (gerçek testin toplam saati) kaydedin.

Doğrulama Testi için Çıkış Ölçütleri

- Tüm test senaryoları yürütüldü.
- Tüm Yazılım Problem Raporlama'lar tatmin edici bir şekilde çözülmüştür. (Çözünürlük, hataların düzeltilmesi, daha sonraki bir yayına ertelenmiş, hata olmamasına karar verilmiş vb. İçerebilir). Tüm taraflar bu kararı kabul etmelidir. Bu kriter, tüm yüksek öncelikli hataların sabitlenmesi gerektiğini belirtirken, düşük öncelikli hataların durum bazında ele alınabileceğini belirtmek için tanımlanabilir.
- Yazılım Problem Raporlama'lar sonucunda yapılan tüm değişiklikler test edilmiştir.
- Yazılımla ilişkili tüm belgeler (Sistem Gereksinim Tanımlama, Yazılım Tasarım Tanımları, test belgeleri gibi) doğrulama testi sırasında yapılan değişiklikleri yansıtacak şekilde güncellendi.
- Test raporu gözden geçirildi ve onaylandı.

6.9. Test Planlama

Test Planlama

- Test Planı - Yapılacak işin kapsamını tanımlar
- Test Prosedürü - yürütülecek tüm bireysel testleri (test komut dosyaları) tutan bir kapsayıcı belgedir
- Test Raporu - test komut dosyaları çalıştırıldığında neler olduğunu belgeler

- Cevaplanacak sorular:
  - Kaç test gerekli?
  - Bu testleri geliştirmek ne kadar sürer?
  - Bu testleri yapmak ne kadar sürer?

- Ele alınacak konular:
  - Test tahmini
  - Test geliştirme ve resmi doğrulama
  - Doğrulama hazırlık incelemesi ve resmi doğrulama
  - Test tamamlama kriterleri

Test Tahmini

- Gerekli test vakalarının sayısı şu şekildedir:
  - Yazılım Gereksinim Tanımlamadaki tüm fonksiyonların ve özelliklerin test edilmesi
  - Aşağıdakiler dahil olmak üzere uygun sayıda Bir Müşteri Gibi testi:
    - Yanlış yap
    - Yanlış veya hatalı giriş kombinasyonları kullanın
    - Yeterince yapma
    - Hiçbir şey yapma
    - Çok fazla yap
  - Bazı test kapsamı hedefine ulaşması
  - Yazılım güvenilirliği hedefine ulaşması

Test Tahminde Dikkat Edilecek Noktalar

- Test Kompleksliği - Birkaç büyük Sistem Tasarım Tanımı daha küçük olanlara sahip olmak daha iyidir.
- Farklı Platformlar - Farklı platformlar, işletim sistemleri vb. Için testlerin değiştirilmesi gerekiyor mu?
- Otomatik veya Manuel Testler - Otomatik testler geliştirilecek mi? Otomatik testler oluşturmak için daha fazla zaman alır, ancak çalıştırmak için insan müdahalesi gerektirmez.

Test prosedürü

- Test senaryolarının toplanması
- Her test komut dosyasının ayrılmaz bir parçası beklenen sonuçtur
- Test Prosedürü dokümanı, testlerin daha kolay tekrarlanabilmesi için her testin kesintisiz ve temiz bir kopyasını içermelidir.

6.10. Testlerin Sonuçlarının Raporlanması

Test sonuçları Yazılım Test Raporları ile raporlanmakta ve ilgili kişilere duyurulmaktadır. Testler sırasında tespit edilen hatalar, Hata Takip aracına girilmekte ve problemin ilgili kişiye atanmasından doğrulanmasına kadar bu araç üstünden takibi gerçekleştirilmektedir. Hataların kayıt altına alındığı Hata Takip araçları raporlama konusunda birçok olanak sunmaktadır. Hata Takip aracına girilen bir yazılım hatası, yazılım sorumlusu tarafından incelenmekte, yazılımda gerekli düzeltme yapıldıktan sonra girilen hatanın düzeltildiği aynı araç yoluyla bildirilmektedir. Yine aynı araç üzerinden hatanın düzeltildiği bilgisini alan test sorumlusu hatanın gerçekten düzeltildiğini yazılımın yeni sürümü üzerinde ilgili testi tekrar gerçekleştirerek hatanın giderilip giderilmediğini doğrulamaktadır. Özetle, test sorumlusu ile yazılım sorumlusu arasındaki hata takibi, Hata Takip aracı kullanılarak yapılmaktadır. Böylelikle hataların sağlıklı bir şekilde kayıt altına alınması sağlanmaktadır.

Yeterlilik testlerinin tamamlanmasının ardından test kayıtları kullanılarak Yazılım Test Raporu hazırlanır. Test raporları Yazılım Konfıgürasyon Birimi sorumluları ve kalite temsilcileri tarafından incelenerek Yazılım Konfıgürasyon Birimi yeterlilik testinin tamamen yapılıp yapılmadığı, Yazılım Konfıgürasyon Birimi'nin yeterliliği ve Sistem/Alt Sistem Entegrasyon ve Test aşamasına hazır olup olmadığı değerlendirilir.

Test raporu

- Her bir test komut dosyasının tamamlanmış kopyası, icra edildiği kanıtlanır (yani testi yapan kişinin imzasıyla)
- Her Yazilim Problem Raporlama çözüldüğünü gösteren kopyasıdır.
- Açık veya çözümlenmemiş Yazilim Problem Raporlama listesi
- Her taban çizgisinde bulunan Yazilim Problem Raporlama'lerin tanımlanması, her bir taban çizgisinde toplam yazilim problem raporlama sayısı
- Her bir yazılım taban çizgisi için yürütülen regresyon testleri

Doğrulama Test Planı, IEEE – Standard 1012-1998

1. Genel bakış
    a. organizasyon
    b. Görevler ve Takvimler
    c. Sorumluluklar
    d. Araçlar, Teknikler, Yöntemler
2. Süreçler
    a. Yönetim
    b. Edinme
    c. Arz
    d. Gelişme

    e. Operasyon
    f. Bakım
3. Raporlama Gereksinimleri
4. İdari Gereksinimler
5. Belge Gereksinimleri
6. Kaynak Gereksinimleri
7. Tamamlama Kriterleri

7. CMMI ve Sertifika seviyeleri

1989 yılında Carnegie Mellon Üniversitesi, ABD Savunma Bakanlığındaki Yazılım Mühendisliği Enstitüsü (SEI) ile "Capability Maturity Model - CMM" olarak adlandırılan normları geliştirmişlerdir. Bu normlar, bir yazılım sürecini tanımlayan disiplinli bir sürece giden bir yol haritasını çizer. ***Temel prensip karşılaştırma yapan yeteneğin zamanla gelişerek ölçülebileceği üzerinedir.***

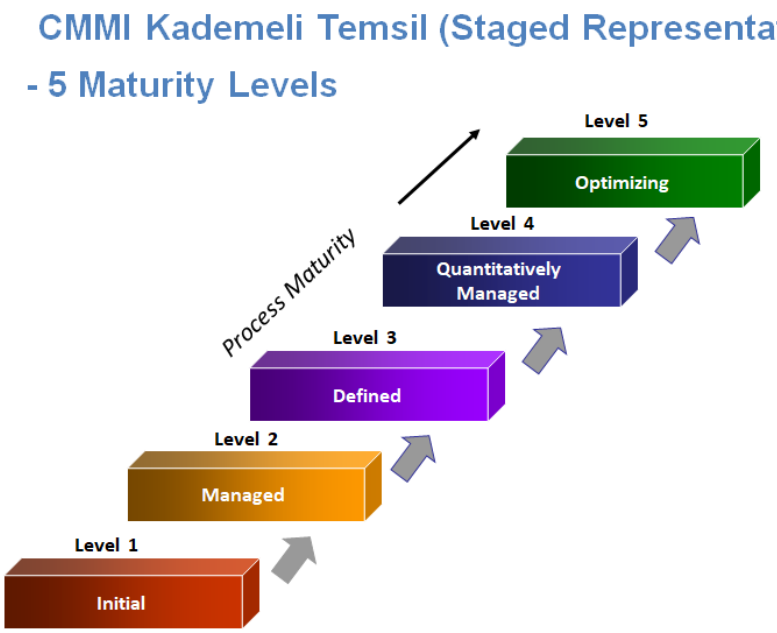
SEI, yazılımın yanında farklı alanlar için CMM normları tanımlamıştır,

- Yazılım mühendisliği
- Tümleşik ürün geliştirme
- Sistem mühendisliği
- Tedarik yönetimi.

CMMI, performans arttırmaya yönelik etkili bir süreç iyileştirme yaklaşımıdır. CMMI, kurumların organizasyonel fonksiyonlarının entegre edilmesine, süreç iyileştirmede amaç ve önceliklerinin belirlenmesine, kalite iyileştirme süreçlerine rehberlik edilmesine ve mevcut süreçlerin değerlendirilmesine yönelik bir referans yaratmayı hedefler.

CMMI genellike IT sektöründe ve özellikle yazılım geliştirme konularında uygulamaya yönelik referans normlar geliştirmektedir. Özellikle savunma sanayi başta olmak üzere; Ar-Ge ve yeni ürün geliştirme konusunda hizmet veren kurumlarısertifika etmektedir.

Sertifika Seviyeleri



Seviye 1 : Initial - Baslangıç
Yazılım geliştirme süreci, genel olarak kaotik bir ortamda yol almaktadır. Bu yolda amaca doğru ilerlerken başarı ancak kişisel çabalar ile sağlanmaktadır. Herhangi bir kriz durumunda, olsa bile tüm planlar unutulur, hatırlanmaz; yok sayılır. Destek sağlanacak yetenekler, beyin fırtınaları önceden düşünülmez.

Seviye 2 : Repeat - Tekrarlama
Projeler planlıdır, başarı adım adım gelir. Süreçler ölçülür ve kontrol edilir. İş dağılımı yapılmış, ekip birbirleri ile haberdar, sağlıklı bir iletişim ortamı oluşturulmuştur. Yazılım süreçindeki tüm işlevlerin gereksinimleri, çıktıları yönetilmektedir. Projedeki tüm paydaşların talep ve değişiklik istekleri göz anında alınmakta ve değişiklikler sürekli revize edilerek işlevsellik sağlanmaktadır.

Seviye 3 : Define - Tanım
Tüm süreçler standartlar, prosedürler, normlar ile açıklanır. Standartlaşan süreç işlevleri organizasyonel boyutta kabul görmüş ve tutarlıdır. Projeler, yaşayan organizmalara dönüştürülmüştür. Tüm işlevler özenli ve detaylı olarak tanımlanmıştır.

Seviye 4 : Manage - Yönetme
Süreçler ve işlevler istatistiksel ve diğer ölçüm teknikleri ile kontrol edilir. Kalite ve süreç performansı yönetimde bir kriter olarak kullanılır. Ölçeklendirilmiş hedefler, paydaşların ihtiyaç ve istekleri, temel alınarak belirlenir.Süreçler ölçülür ve zarfsal yani genişliği değişken sınırlar içerisinde gerçek veriler baz alınarak öngörüde bulunulur.

Seviye 5 : Optimize –İyileştirme
Hassas işlevlerin belirlenmesi, özellikle kaotik davranışı başlatacak hassas süreçlerde, sayısal geri beslemeler ve teknolojilerden yararlanılarak sürekli iyileştirilme yapılır. Tüm organizasyon süreç iyileştirmeye odaklanmıştır. Yazılım geliştirme süreç yeteneği "Sürekli İyileşen" olarak tanımlanabilir.

4.seviye, süreç değişkenliğinin nedenleri ve sonuçları istatistiksel öngörüsü ile denetlenirken 5.seviyede süreç değişkenliğinin nedenlerini bulma ile ilgilenir.

7.1. CMMI süreç alanları

Kurumsal Süreç Yönetimi

- Odaklanma
- Tanımlama
- Eğitim
- Performansı
- Yaratıcılık ve Yaygınlaştırma
- Hafıza

Proje Yönetimi
Proje Planlama
Proje İzleme ve Kontrol
Tedarikçi Anlaşma Yönetimi

Bütünleşik Proje Yönetimi
Risk Yönetimi
Bütünleşik Takım
Bütünleşik Tedrikçi Yönetimi
Sayısal Proje Yönetimi

Mühendislik
RD Gereksinim Geliştirme
Gereksinim Yönetimi
Teknik Çözüm
Ürün Tümleştirme
Doğrulama
Geçerleme

Destek
Konfigürasyon Yönetimi
Süreç ve Ürün Kalite Güvencesi
Ölçme ve Çözümleme
Karar Analizi ve Çözüm
Nedensel Analiz ve Çözüm

7.2. Kategori Süreç Alanları

Yönetim Süreçleri
Bütünleşik Kurum Yönetimi
Proje Yönetimi
Risk Yönetimi
Tedarikçi Anlaşma Yönetimi
Bütünleşik Takım

Yaşam Döngüsü Süreçleri
İhtiyaçlar
Gereksinimler
Tasarım
Gerçekleştirim
Entegrasyon
Kurulum ve Dağıtım
İşletme ve Destek
Değerlendirme

Destek Süreçler
Dış Kaynak Kullanımı
Seçenek Analizi
Ölçme ve Çözümleme
Kalite Güvencesi ve Yönetimi
Konfigürasyon Yönetimi
Bilgi Yönetimi
Süreç Tanımlama
Süreç İyileştirme
Eğitim
Yenilik

7.3. SSE-CMM süreç alanları

Proje ve Organizasyon Süreçleri
Kaliteyi Sağla
Konfigürasyonu Yönet
Proje Riskini Yönet
Teknik Eforu İzle ve Kontrol Et
Teknik Eforu Planla
Organizasyonun Sistem Mühendisliği Sürecini Tanımla
Organizasyonun Sistem Mühendisliği Sürecini İyileştir
Ürün Hattı Değerlendirmesini Yönet
Sistem Mühendisliği Destek Ortamını Yönet
Yetenek ve Bilginin Sürekliliğini Sağla
Tedarikçilerle Koordinasyon Sağla

7.4. Güvenlik Mühendisliği Süreç Alanları

Mühendislik Güvenlik Gereksinimlerini Belirle
Güvenlik Girdileri Sağla
Güvenlik Durumunu İzle
Güvenlik Kontrollerini Yönet
Güvenliği Sağla

Güvence Güvenliği Doğrula ve Geçerle
Güvenlik Argümanı Oluştur
Risk Tehditleri Belirle
Sistem Açıklarını Belirle
Etkilerini Tanımla
Güvenlik Riskini Belirle

7.5. Yazılım Garanti Olgunluk Modeli (SAMM)

Yönetim (Governance): Organizasyonda uygulanacak yazılım güvenliği programı, güvenlik çalışmalarının performansını ölçme yöntemleri, belirlenmiş standartlara uyum sağlanması ve çalışanların yazılım güvenliği konusunda eğitilmesi gibi uygulamaları kapsamaktadır.

Yapım (Construction): Güvenli yazılım geliştirmek için yazılım gereksinimi ve tasarımı aşamalarında gerçekleştirilmesi gereken güvenlik eylemlerine değinmektedir. Yazılımın karşılaşacağı tehditlerin değerlendirilmesi, güvenlik ihtiyaçlarının belirlenmesi ve güvenli mimarinin oluşturulması gibi güvenlik uygulamalarını içerir.

Doğrulama (Verification): Tasarım, yazılım kodlama ve yazılım testleri aşamasında gerçekleştirilmesi gereken güvenlik gözden geçirmelerini ve güvenlik testlerini kapsamaktadır. Tasarım gözden geçirmesi, kod analizi ve güvenlik testleri bu kapsamdaki güvenlik uygulamalarıdır.

Uygulama (Deployment): Yazılımın piyasaya sürülmesi ve desteğinin verilmesi faaliyetlerini kapsamaktadır. Sistem açığı yönetimi, ortam sıkılaştırması ve operasyonel bilgi aktarımı bu aşamadaki güvenlik uygulamalarıdır.

7.6. SAMM yapısı

Yönetim Yapım Doğrulama Uygulama
Strateji ve metrikler Tehditdeğerlendirmesi
Tasarım gözdengeçirme
Açıklıkyönetimi
Politika ve uyum Güvenlikgereksinimleri
Kod gözden geçirme Ortamsıkılaştırması
Eğitim ve rehberlik Güvenli mimari Güvenlik testi Operasyonel
Bilgiaktarımı

Yapısal Doğruluk
Yapısal doğruluk (Correctness by Construction), Praxis High Integrity Systems tarafından yüksek bütünlük gerektiren yazılımları geliştirmek için oluşturulan bir metodolojidir.
Bu metodoloji, emniyet ve güvenlik bakımından kritik önem taşıyan sistemleri büyük bir başarıyla geliştirmek için kullanılır.
Correctness by Construction modeli, yazılımın güvenlik ve emniyet özelliklerini belirtmek için neredeyse her zaman resmi bir metot kullanır.
Bu model aşağıdaki ilkeleri benimser:
1. Değişen gereksinimlere ayak uydurma,
2. Hem hatasız yazılım hem de doğrulama (verification) için test,
3. Test etmeden önce hataları giderme,
4. Doğrulaması kolay olacak şekilde yazılım geliştirme,
5. Aşamalı geliştirme,
6. Kullanıcı kılavuzu, iş süreçleri, tasarım dokümanları, yorumlarla desteklenmiş kaynak kod ve test durumları oluşturma.
Correctness by Construction, resmi yöntemleri geliştirme aktivitelerine dâhil eden sayılı güvenli yazılım geliştirme süreçlerinden bir tanesidir.
Correctness by Construction ile geliştirilen yazılımlarda daha az resmi yaklaşımlarla geliştirilen sistemlere göre hata oranı azalmıştır. Her KLOC için 0.04 hata oranıyla, endüstri ortalamasına göre çok daha iyi başarı elde etmiştir.

8. Sonuç

Şelale modeli yazılım mühendisliğinin en eski ve temel modelidir. Günümüzde kamu kuruluşları ve büyük şirketler tarafından her türlü projenin yönetim standardı olarak kabul görmektedir. Çevik yöntemler ise geleneksel yöntemlerin yetersiz kaldığı değerlendirilen konular için alternatif çözümler olarak ortaya çıkmıştır.

Şelale modeli projenin geçmiş süreçlerina bağımlı olarak yürütüldüğü için anlayış olarak geriye dönüktür ve bütün proje elemanlarının kendisi için tanımlanan işi yapmasınınbeklendiği bir mühendislik düzenidir. Bu düzenden faydalanan şelale modeli daha çok, gereksinimlerin ve hedeflerin açık ve net olduğu, çözümlerin ve yapılacak işin bilindiği, gereksinimlerin değişmeyeceği, üretilecek yazılımın hata toleransının olmadığı ve fonksiyonel bütünlüğe sahip projeler için uygundur. Bu özellikleri nedeniyle en iyi kullanım alanı olarak savunma sistemleri ve gömülü sistemler verilebilir. Çevik yöntemler ise değişimi ve geri beslemeyi teşvik eden yapısı nedeniyle ileri dönük bir anlayışa sahiptir. Otonom takımların her işi kendi kendine yapması ve bu süreçte çapraz fonksiyonellikten ve deneycilikten faydalanmaları beklenir. Bu da düzenden çok bir yaratıcılık yaklaşımı gerektirir. Çevik yöntemler gereksinimlerin ve hedeflerin belirsiz ve bilinmez olduğu, değişikliğin sık, değer üretiminin kısa vadede arzu edildiği projeler için daha uygundur. Bu nedenle çevik yöntemler için en iyi kullanım alanı web tabanlı iş akış yönetimi sistemleridir.

İki yöntemin karışımı olarak ortaya çıkan metodolojiler de günümüzde popülerlik kazanmaya devam etmektedir. Örneğin birçok organizasyon, yazılım projesinin kodlama ve test süreçlerinın scrum metodu ile, planlama, tasarım ve teslimat gibi geri kalan süreçlerin da şelale modelinin prensipleri ile gerçekleştirildiği Water-Scrum-Fall metodolojisini uygulamaktadır.

Günümüzde teknolojideki hızlı gelişim, bütün iş süreçlerini, pazar dinamiklerini ve bu doğrultuda yazılıma ihtiyaç duyan müşterilerin gereksinimlerini etkilemekte ve geçmişe nazaran daha değişken bir ortam yaratmaktadır. Bu değişken ortamda çevik yöntemler şelale modeline göre avantajlı olsa da bazı şartlarda şelale modelinin sağladığı güven ve düzeni sağlayamamaktadır. Bu nedenle başarılı bir yazılım proje yöneticisi hem şelale modeli hem de çevik yöntemlerin anlayışına, avantaj ve dezavantajlarına hâkim olmalı ve projenin ihtiyaçlarına göre uygun bir metodoloji benimsemelidir. Bu çalışma yazılım proje yönetiminde karşılaşılabilecek durumlar için şelale modeli ve çevik yöntemlerin avantaj ve dezavantajları açısından bir öneri niteliği taşımaktadır. Şartlara göre bir metodoloji seçimi yapılmalı ya da ihtiyaçlara göre metodolojilerin arzu edilen teknikleri birleştirilerek karma bir metodoloji oluşturulmalıdır.

9. Kaynaklar (References)

1) Software Engineering, I. Sommerville, Pearson Education Inc., A.B.D., 2011.
2) Report on Software Engineering Conference, NATO Science Committee, P. Naur, B.Randell, Almanya, 1968.
3) Yazılım Proje Yönetimi: Şelale Modeli ve Çevik Yöntemlerin Karşılaştırılması, Cevriye GENCER1, Ali KAYACAN, BİLİŞİM TEKNOLOJİLERİ DERGİSİ, CİLT: 10, SAYI: 3, TEMMUZ 2017.
4) Yazılım Mühendisliği,Yrd.Doç.Dr. Yunus Emre SELÇUK, Şubat 2017.
5) **Sistem/Yazılım Geliştirme Sürecinde Doğrulama Faaliyetleri,** Mikrodalga ve Sistem Teknolojileri (MST) Grubu, ASELSAN A.Ş., Ankara.