

ÇEVİK YAZILIM GELİŞTİRME

“AGILE”

Sürekli yazılım geliştirme

- İş çevrelerinin hızla değişmesinden dolayı şirketler yeni fırsatlara ve rekabete cevap verebilmelidir
- Bu yazılımı ve çabuk geliştirmeyi gerektirir, ancak teslimat yazılım sistemleri için her zaman en kritik gereksinim değildir
- Şirketler eğer yazılımın temel fonksiyonlarıyla çabuk teslimatı mümkünse düşük kaliteli yazılıma razı olabilirler

Gereksinimler :

- Değişen çevreden dolayı, genellikle kararlı, istikrarlı sistem gereksinimi ayarlarına ulaşmak mümkün değildir
- O yüzden waterfall modeli kullanılan geliştirme elverişsizdir, tekrarlanan tanımlama ve teslimat geliştirme tabanlı yaklaşım yazılımı hızlı teslim etmenin tek yoludur.

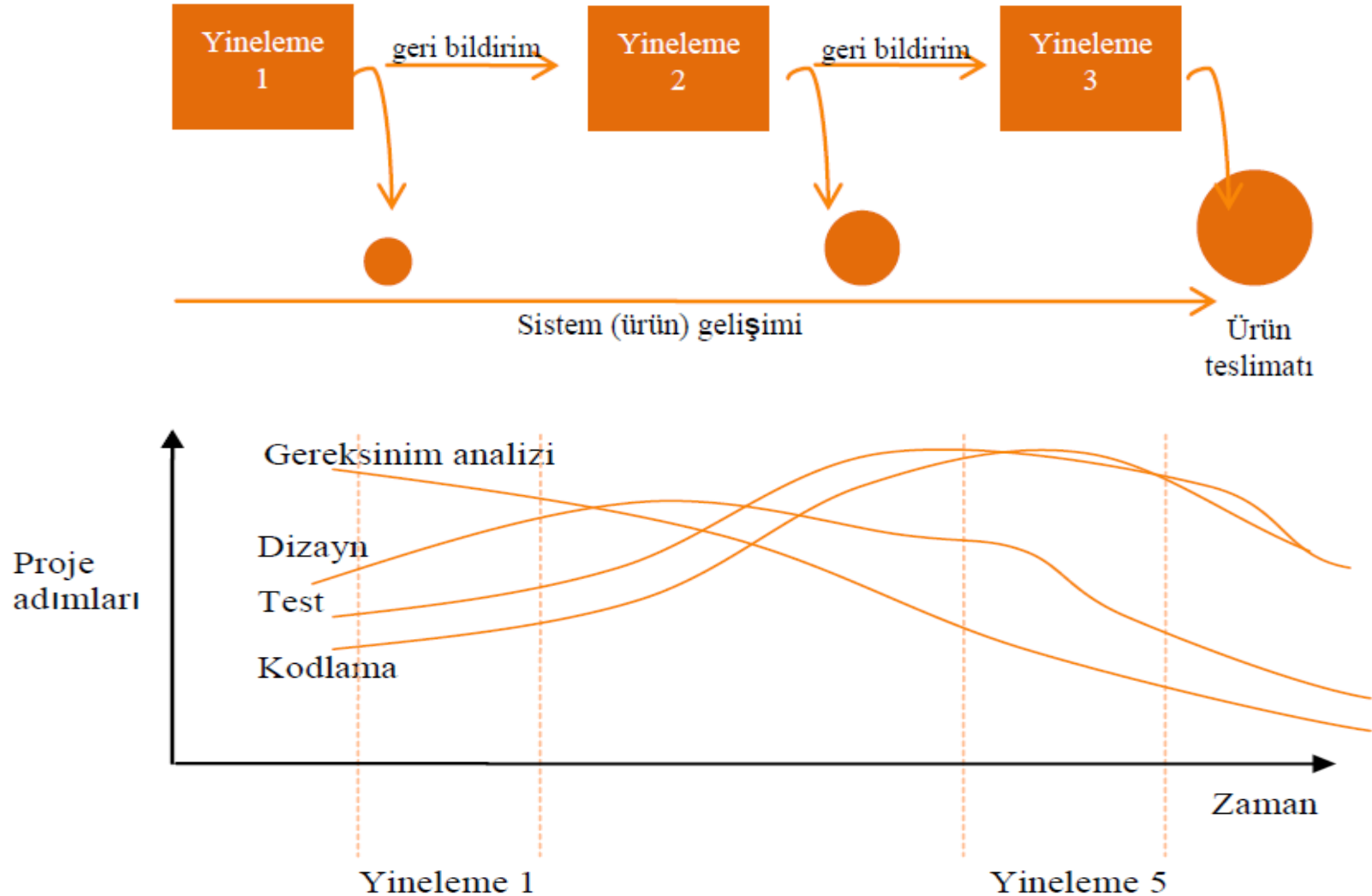
Özellikleri :

- Tanımlama (specification), tasarım (design) ve gerçekleştirme (implementation) süreçleri eş zamanlıdır. Detaylı tanımlama yoktur ve tasarım dokümantasyonu minimize edilmiştir.
- Sistem bir sıra ekleme şeklinde geliştirilir. Son kullanıcılar her artışı dener ve daha sonraki ekleme için teklifte bulunurlar.
- Sistem kullanıcı ara yüzü genellikle bir etkileşimli geliştirme sistemi kullanılarak geliştirilir.

Artışlı geliştirmenin avantajları

- **Müşteri servislerinin hızlandırılmış teslimatı**
Her artış müşteriye en öncelikli işlevselliği gönderir
- **Sistemle kullanıcı bağlantısı.** Kullanıcılar geliştirme sürecinin içine dahil edilmelidir, zira sistem daha büyük bir olasılıkla onların ihtiyaçlarını karşılar ve kullanıcılar daha fazla sisteme adanır.

TEKRARLANAN YAZILIM GELİŞTİRME METODU



verimlilik

esneklik



Tekrarlanan Yazılım Geliştirme

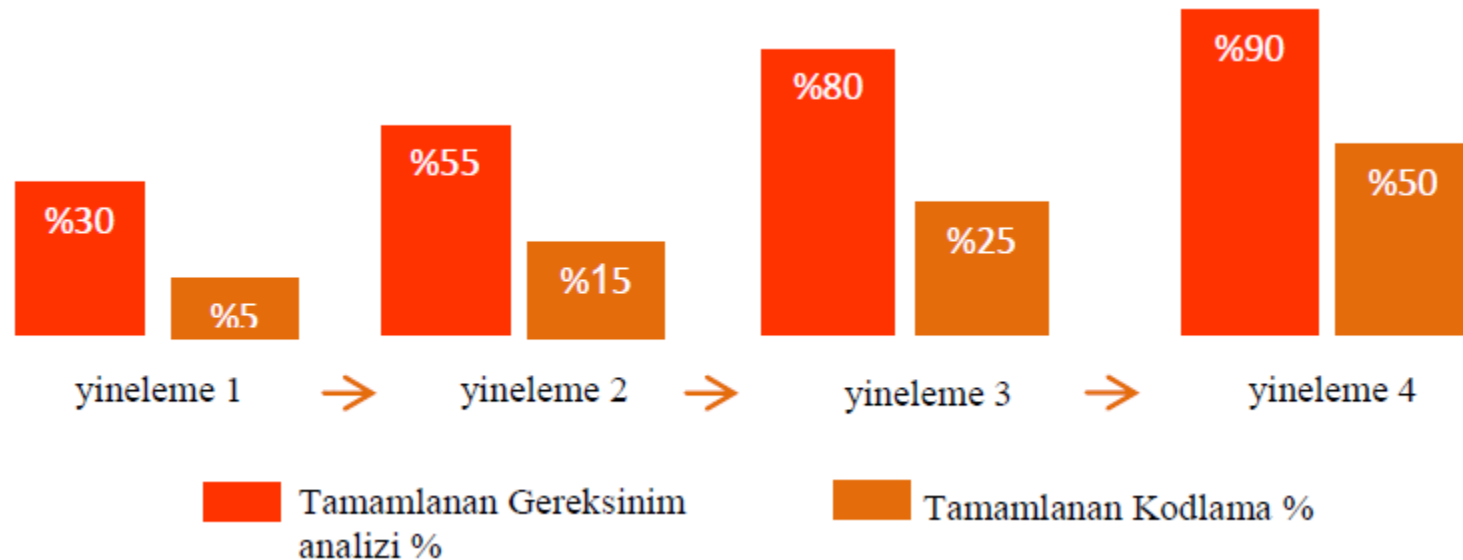


başarı

Artışlı (incremental) geliştirmenin problemleri

- **Yönetim problemleri**
 - İlerleme, hüküm verme konusunda zor olabilir ve problemleri bulmak zordur çünkü neyin yapıldığını gösteren bir dokümantasyon yoktur.
- **Sözleşme problemleri**
 - Normal bir sözleşme bir tanımlama (specification) içerebilir; tanımlamanın olmaması halinde farklı sözleşme şekilleri kullanılmalıdır.
- **Onaylama (validation) problemleri**
 - Tanımlama yoksa, sistem neye göre test edilecek?
- **Bakım (maintenance) problemleri**
 - Sürekli değişim yazılımı yeni gereksinimleri karşılamak için yapılan gelişme ve değişmeyi daha pahalı hale getirerek yazılımın yapısına zarar verme eğilimindedir.

- Ana projeyi parçalara bölerek karmaşıklığı azaltmakta,
- Bünyesinde bulundurduğu geri bildirimler sayesinde değişimi desteklemekte,
- Riskleri azaltmakta,
- Projelerin başarı oranını arttırmaktadır.



İlk örneği oluşturma (Prototyping)

- Bazı büyük sistemler için, artıslı(incremental) tekrarlanan geliştirme ve teslimat elverişsiz olabilir; bu özellikle birçok takımın farklı yerlerde çalışması durumunda doğrudur.
- Prototipleme, gereksinimleri formüle etmeye temel olarak bir deneysel sistem geliştirildiğinde kullanılabilir. Bu sistem şartname konusunda anlaşmaya varıldığında atılır.

Çelişkili amaçlar

- Artışlı (incremental) geliştirmenin amacı son kullanıcılara çalışır bir sistem teslim etmektir. Geliştirme en iyi anlaşılan gereksinimlerle başlar.
- Prototiplemeyi atmanın amacı sistem gereksinimlerini onaylamak veya türetmek içindir. Prototipleme süreci en az anlaşılan gereksinimlerle başlar.

Çevik metodlar

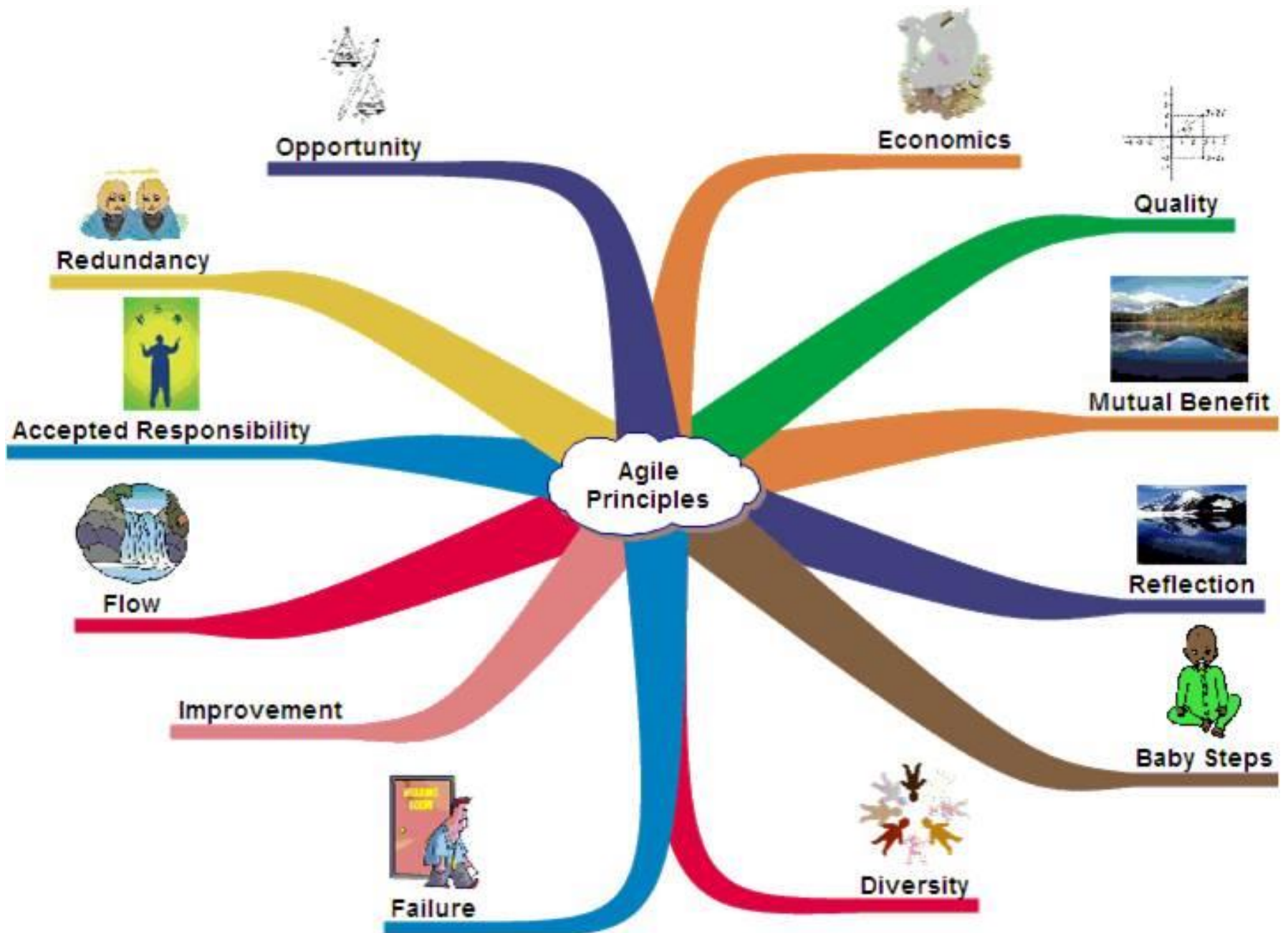
- Dizayn metotlarındaki genel masraflardan kaynaklanan hoşnutsuzluk çevik metotların oluşturulmasına sebep olmuştur. Bu metotlar:
 - Dizayn yerine kod üzerine odaklanma
 - Yazılım geliştirmede tekrarlanan bir yaklaşımı temel alma;
 - Çalışan yazılımı çabuk teslim etme ve bunu değişen ihtiyaçları karşılamak için çabuk geliştirmeyi planlama;
- Çevik metotlar büyük olasılıkla küçük/orta ölçekli şirket sistemleri veya PC ürünleri için idealdir.

ÇEVİK YAZILIM GELİŞTİRME METODU

Çevik yazılım süreçleri, 1950'lerdeki üretim alanında verimliliğin artırılması için geliştirilen yalın yaklaşımların, yazılım sektöründe bir uzantısı olarak ortaya çıkmıştır. Yazılım dünyasında çeşitli çevik yaklaşımlara 1970'lerden itibaren rastlanabilmekle birlikte, çevik yazılım metodolojilerinin kullanımı 1990'larda hız kazanmış ve geçtiğimiz son 7-8 yıl içerisinde de tüm dünyada başarılarını kanıtlayarak popüleritesini arttırmıştır. Şu anda, dünyadaki birçok yazılım şirketinde ve birçok yazılım projesinde yazılımlar, çevik yaklaşımlarla geliştirilmektedir.

Çevik yazılım geliştirme metodu, tekrarlanan yazılım geliştirme metodu taban alınarak geliştirilmiş, sık aralıklarla parça parça yazılım teslimatını ve değişikliği teşvik eden bir yazılım geliştirme metodolojisidir. Çevik geliştirme;

- değişimi,
- takım içerisindeki iletişimin arttırılmasını,
- parça parça yazılım teslimatını,
- test odaklı yazılım geliştirilmesini,
- ve uyumlu planlamayı teşvik eder.



Çevik Yazılım Geliştirme Manifestosu

2001 yılında, dünyanın önde gelen çevik yazılım geliştiricileri (XP, Scrum gibi metodolojilerin yaratıcıları) ortak bir zeminde buluşabilmek adına bir araya gelerek “çevik yazılım geliştirme manifestosu” nu ve “çevik yazılımın prensipleri” ni yayınlamışlardır. Böylelikle çevik metotların projelere genel bakış açıları ifade edilmiştir.

Bu manifestoda;

Bireyler ve aralarındaki etkileşimlerin, kullanılan araç ve süreçlerden;

Çalışan yazılımın, detaylı dokümantasyondan;

Müşteri ile işbirliğinin, sözleşmedeki kesin kurallardan;

Değişikliklere uyum sağlayabilmenin, mevcut planı takip etmekten;

daha önemli ve öncelikli olduğu belirtilmektedir.

Çevik Yazılımın Prensipleri:

- 1- İlk öncelik, sürekli, kaliteli yazılım teslimatıyla müşteri memnuniyetini sağlamaktır.
- 2- Proje ne kadar ilerlemiş olursa olsun değişiklikler kabul edilir. Çevik yazılım süreçleri değişiklikleri müşteri avantajına dönüştürürler.
- 3- Mümkün olduğunca kısa zaman aralıklarıyla (2-6 hafta arası) çalışan, kaliteli yazılım teslimatı yapılır.
- 4- Analistler, uzmanlar, yazılımcılar, testçiler vs. tüm ekip elemanları bire bir iletişim halinde, birlikte çalışırlar.
- 5- İyi projeler motivasyonu yüksek bireyler etrafında kurulur. Ekip elemanlarına gerekli destek verilmeli, ihtiyaçları karşılanarak proje ile ilgili ekiplere tam güvenilmelidir.

Çevik Yazılımın Prensipleri:

6- Ekip içerisinde kaliteli bilgi akışı için yüz yüze iletişim önemlidir.

7- Çalışan yazılım, projenin ilk gelişim ölçütüdür.

8- Çevik süreçler mümkün olduğunca sabit hızlı, sürdürülebilir geliştirmeye önem verir.

9- Güçlü teknik alt yapı ve tasarım çevikliği arttırır.

10- Basitlik önemlidir.,

11- En iyi mimariler, gereksinimler ve tasarımlar kendi kendini organize edebilen ekipler tarafından yaratılır.

12- Düzenli aralıklarla ekipler kendi yöntemlerini gözden geçirerek verimliliği arttırmak için gerekli iyileştirmeleri yaparlar.

Gereksinimlerdeki Değişiklikler

Proje Büyüklüğü

Çevik Metodolojilerin Getirileri:

Düşük Risk:

Tekrarlanan yazılım geliştirme metotları, proje risklerini azaltıp başarıyı arttırmakta ve hata oranlarını düşürerek verimliliği yükseltmektedir.

Bunun arkasında yatan en temel etken, daha projenin başlarında geliştirilen program parçacıkları sayesinde, proje ekibinin yetkinliklerinin ve projede kullanılan her türlü araçların önceden denenerek eksiklerinin görülebilmesidir.

Ayrıca, parçalı geliştirme süreci içerisinde proje hızlı olarak şekillenmekte ve proje başlangıcında fark edilemeyen riskler ciddi sorunlara yol açmadan önce görülebilir hale gelmektedir.

Çevik Metodolojilerin Getirileri:

Değişimin Teşvik Edilmesi:

Yazılım projelerinde değişiklik kaçınılmazdır. Orta çaplı projeler dahi proje süresince başlangıçlarına göre %30 oranlarında değişime uğramaktadır.

Bu nedenle, değişim yazılım projelerinin doğasıdır ve bu gerçeklik çevik metodolojilerin de üzerinde önemle durduğu bir etkidir.

Çevik metodolojiler değişime karşı gelmek yerine değişimi müşteri avantajına dönüştürmeye yönelik olarak çalışırlar.

Çevik metodolojiler, önerdiği parçalı yazılım üretimi ve her adımdaki güçlü bilgi alış verişiyle, değişim gereksinimlerinin mümkün olduğunca projelerin başlangıç adımlarında fark edilmesini ve projenin değişime hızlı bir şekilde adapte olabilmesini sağlarlar.

Çevik Metodolojilerin Getirileri:

Karmaşıklığın Yönetimi:

Yazılım projelerinin hacimleri büyüdükçe karmaşıklıkları artar ve buna bağlı olarak hata oranları da artar.

Çevik yöntemler ise projeleri, daha kolay yönetilebilir küçük parçalara bölerek ele alırlar.

Böylece, projelerin büyüklüğü ne olursa olsun, küçük parçalara ayrılarak ele alınan projeler için karmaşıklık en düşük seviyeye indirilir.

Çevik Metodolojilerin Getirileri:

Sürekli Yazılım Teslimi:

Çevik metodolojilerde her yineleme sonunda çalışan bir programcık meydana getirilmektedir.

Projenin başlarından itibaren devam ederek büyüyen bu parçacıklar, müşterilere elle tutulur aşamalar olarak sunulmakta ve bu da müşteri memnuniyetini arttırmaktadır.

Yapılan araştırmalar, müşterilerin tamamlanmış ama çalışmayan programlar yerine, çalışan durumda ama tamamlanmamış programları tercih ettiklerini göstermiştir.

Çevik Metodolojilerin Getirileri:

Yüksek Kalite:

Tekrarlanan yazılım geliştirme metotlarının bünyesinde, test odaklı yazılım geliştirme mantığı bulunmaktadır.

Proje başından başlayarak tüm test süreçlerinin yazılım geliştirme süreci içerisinde beraber yürütülmesi sayesinde, hatalar büyümeden fark edilerek hızlı bir şekilde düzeltilebilmektedir.

Proje hacimleri büyüdükçe hata oranları da artmaktadır. Ancak çevik metodolojiler, projeleri parçalara bölerek ve her bir parçayı kendi gelişimi içerisinde de test ederek hata oranlarının düşürülmesini sağlamaktadır.

Çevik Metodolojilerin Getirileri:

Müşteri ihtiyaçlarına daha iyi cevap veren çözümler:

Yinelemeler arasında gerçekleştirilen fikir alışverişleri, çevik yöntemlerin değişikliğe olan yatkınlığı ve müşteri odaklı yaklaşımları sayesinde, projeler, müşteriler ile birlikte değişerek gelişmekte ve proje sonunda da müşteri ihtiyacını en iyi derecede karşılayabilecek programlar ortaya çıkmaktadır.

Müşterilerin çoğunlukla proje başlangıcında tam olarak ne istediklerine dair net birer fikirleri yoktur ve istekler, projenin gelişim süreci ile birlikte şekillenmektedir.

Bu gerçeklik karşısında, çevik yöntemler müşteri odaklı ve esnek yapısıyla, müşteri memnuniyetini en üst seviyede tutmayı başaramaktadırlar.

Çevik metotların problemleri

- Sürece dahil edilen müşterilerin ilgisini sürekli kılmak zor olabilir.
- Takım üyeleri çevik metotları tanımlayan yoğun karışmaya uygun olmayabilirler.
- Önceliklerin değişimi birden fazla stakeholder(ortak) olması durumunda zordur.
- Sadeliği koruma fazladan iş gerektirir.
- Tekrarlanan geliştirmeye farklı yaklaşımlar olduğunda sözleşme bir problem olabilir.