Bölüm 3

Çevik Yazılım Geliştirme (Agile Software Development)

Hedefler

- Çevik yazılım geliştirme metotlarını, çevik manifestoyu ve çevik ve plan- güdümlü geliştirme arasındaki farkları anlamak
- Ekstrem programlamanın temel noktalarını bilmek ve bunları çevik metotların genel prensipleri ile ilişkilendirmek
- Çevik proje yönetiminde Scrum yaklaşımını anlamak
- Çevik geliştime metotlarını büyük yazılım sistemlerine uygularken karşılaşılan sorunlardan haberdar olmak

İçerik

- Çevik Metotlar
- Plan-Güdümlü ve Çevik Geliştirme
- Ekstrem Programlama
- Çevik Proje Yönetimi
- Çevik Metotları Ölçeklendirme

Giriş

- Günümüzde hızlı geliştirim ve teslimat yazılım sistemleri için en büyük taleptir
- Yazılım projelerinde ihtiyaçlar ve talepler sürekli değişim göstermektedir
  - En başta müşteri de ne umacağını bilemez. Teslimattan sonra kullanıcı yorumları gelir
  - Yazılım teslim edildiğinde çoktan eskimiş olabilir
- Gereksinimlerin başlangıçta net bir şekilde belirtildiği geleneksel şelale modelinde yazılımın hızlı üretilmesi beklenemez
- Plan-güdümlü yazılım geliştirme bazı türde projeler için uygundur
  - Yüksek güvenlik gerektiren
- Hızlı değişen iş dünyası, başka ihtiyaçlara sahiptir

Tarihçe

- Hızlı sistem geliştirme ve gereksinimlere uyum sağlamaya olan ihtiyaç uzun süre önce farkedilmiştir
- IBM, artımlı geliştirmeyi 80'lerin başında tanıtmıştır
- 80'lerin başında 4. nesil dillerin de ortaya çıkması, yazılımın hızlı geliştirimi ve teslimatı fikrini desteklemiştir
- Fakat DSDM (Stapleton, 1997), Scrum (Schwaber ve Beedle, 2001), ve ekstrem programlama(Beck, 1999; Beck, 2000) gibi çevik yöntemlerin ortaya çıkışı 90'ların sonunu bulmuştur

Hızlı Sistem Geliştirme

- Birçok yaklaşım var fakat temel olarak şu özellikler gerekli:
  - Gereksinim tanımlama, tasarım ve geliştirim aşamaları dönüşümlü olarak çalışır. Gereksinim dokümanı sadece en temel özellikleri içerir
  - Sistem sürümler şeklinde geliştirilir
  - Kullanıcı arayüzleri kolay ve hızlı bir şekilde interaktif geliştirim sistemi ile tasarlanır
- Çevik metotlar, artımlı geliştirim metotlarıdır, genellikle kullanıcıya iki haftada bir teslim edilen çok küçük parçalardan oluşur
- Kullanıcıyı geliştirme sürecinin içine dahil ederek, değişen gereksinimlerle ilgili çok daha hızlı geri dönüş alınır
- Yazılı belgelerin oluşturulduğu resmi toplantılar yerine kişisel iletişimle belgeler azaltılır

Çevik Metotlar

- 80'le sonu 90'lar başı iyi proje kavramı
- Plan-güdümlü yaklaşımların yükü
  - Çok uzun yıllar süren projeler
  - Takımların durumu
- Plan-güdümlü yaklaşımın küçük çaplı iş uygulamalarında kullanımı
  - İşin büyük bir kısmı gereksinim, planlama vb. ile geçiyor
  - Geliştirme ve teste az zaman ayrılıyor
- 90'larda çevik yazılım geliştirme kavramı ortaya atılıyor

Çevik Metotlar

- Çevik metotlar, yazılımın tanımlanma, geliştirim ve teslim süreçleri için artımlı yaklaşım üzerine geliştirilmiştir
- Müşteriye hızlıca çalışan bir program teslim etmek ve yeni gereksinimleri müşteriden almak hedeflenir
- Hiçbir zaman kullanılmayacak olan belgelerden ve uzun-dönemde belirsiz olan işlerden kaçınılır
- Çevik metotların altında yatan felsefe bu metotların geliştiricileri tarafından çevik yazılım manifestosunda ortaya konmuştur

Çevik Yazılım Manifestosu

Bizler daha iyi yazılım geliştirme yollarını uygulayarak ve başkalarının da uygulamasına yardım ederek ortaya çıkartıyoruz. Bu çalışmaların sonucunda:

- Süreçler ve araçlardan ziyade **bireyler ve etkileşimlere**
- Kapsamlı dökümantasyondan ziyade **çalışan yazılıma**
- Sözleşme pazarlıklarından ziyade **müşteri ile işbirliğine**
- Bir plana bağlı kalmaktan ziyade **değişime karşılık vermeye**

değer vermeye kanaat getirdik.

Özetle, sol taraftaki maddelerin değerini kabul etmekle birlikte, sağ taraftaki maddeleri daha değerli bulmaktayız.

Çevik Metotlar

- En çok bilinen çevik metot Ekstrem Programlama (Beck, 1999; Beck, 2000)
- Scrum (Cohn, 2009; Schwaber, 2004; Schwaber ve Beedle, 2001)
- Crystal (Cockburn, 2001; Cockburn, 2004)
- Adaptive Software Development (Highsmith, 2000)
- DSDM (Stapleton, 1997; Stapleton, 2003)
- Feature Driven Development (Palmer ve Felsing, 2002).

Çevik Metotlar

- Çevik yöntemler iki tip geliştirme için uygundur:
  - Küçük veya orta çaplı bir yazılımlar
  - Müşterinin geliştirme sürecine dahil olacağının sözünü verdiği ve yazılımı etkileyen çok fazla kural veya düzenlemelerin olmadığı yazılımlar

Çevik Yazılım Prensipleri ve Gizli Problemler

- **Müşterinin sürece katılması**
  - Müşterinin bunu istemesi
- **Artımlı teslimat**
- **Süreç değil insanlar**
  - Programcılar sürece ve takıma uygun olmayabilir
- **Değişimi kabul etme**
  - Değişikliklere öncelik verme
- **Basitliği sağlama**
  - Zaman sınırları
- Şirket kültürünü değiştirmek kolay değil
- Sözleşme hazırlamak kolay değil

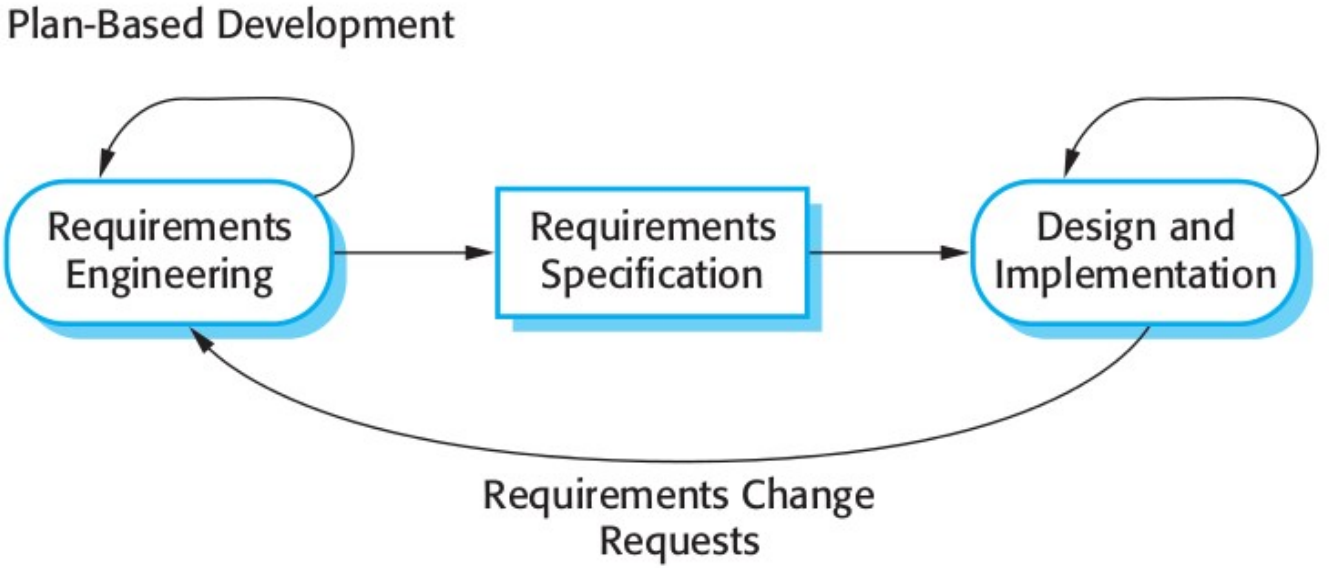
Çevik Yazılımların Bakımı

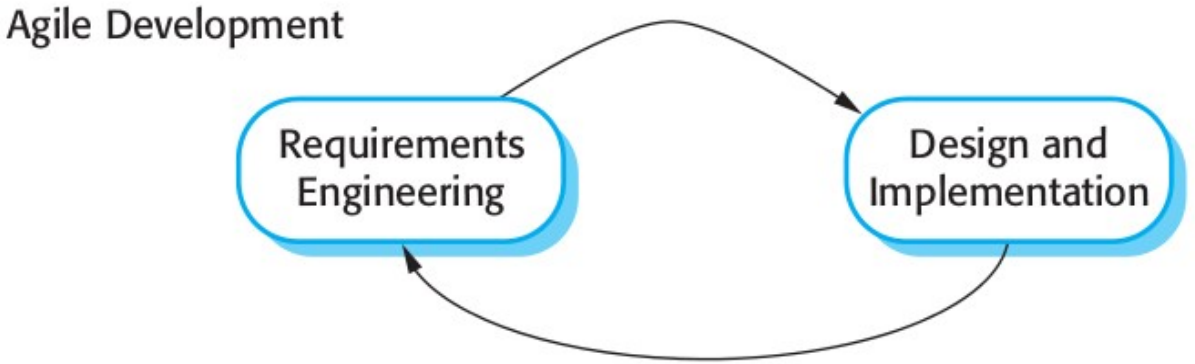
- Dokümantasyonu en aza indirme fikri, bakım sürecini nasıl etkiler?
  - İyi kod yazımı
  - Gereksinim dokümanının önemi
- Çevik metotlarla geliştirilmemiş sistemlerde değişiklikler çevik prensiplere uygun yapılabilir
  - Teslimat sonrası müşteriyi sürece dahil etmek kolay değil
  - Yeni gereksinimler için değişiklik talebi
- Geliştirim ekibindeki değişiklikler
  - Yeni takım elemanlarının sisteme dahil edilmesi

Plan-Güdümlü ve Çevik Geliştirim

- Yazılım geliştirmede çevik yaklaşımlar, tasarım ve gerçekleştirimin yazılım sürecinin en önemli adımları olduğunu kabul eder. Gereksinim belirleme ve test gibi diğer adımlar, tasarım ve geliştirme süreçlerinin kapsamına alınmıştır.
- Buna zıt olarak, plan-güdümlü yazılım mühendisliği, yazılım süreçlerini birbirinden ayrı adımlar olarak tanımlar ve her adımı bir çıktı ile ilişkilendirir. Bir adımın çıktısı bir sonraki süreçte yapılacakları planlamada temel olarak alınır.

Plan-Güdümlü ve Çevik Geliştirim





Plan-Güdümlü ve Çevik Geliştirim

- Plan-güdümlü yazılım süreci artımlı geliştirim ve teslimatı desteklekleyebilir. Gereksinim tanımlama, tasarım ve geliştirimi planlama safhaları örnek olabilir
- Çevik süreç her zaman kod-odaklı olmak zorunda değildir, bazen dokümantasyon da üretilebilir. Yazılım ekibi yeni sürüm çıkarmak yerine sistem dokümantasyonu yapmaya da karar verebilir

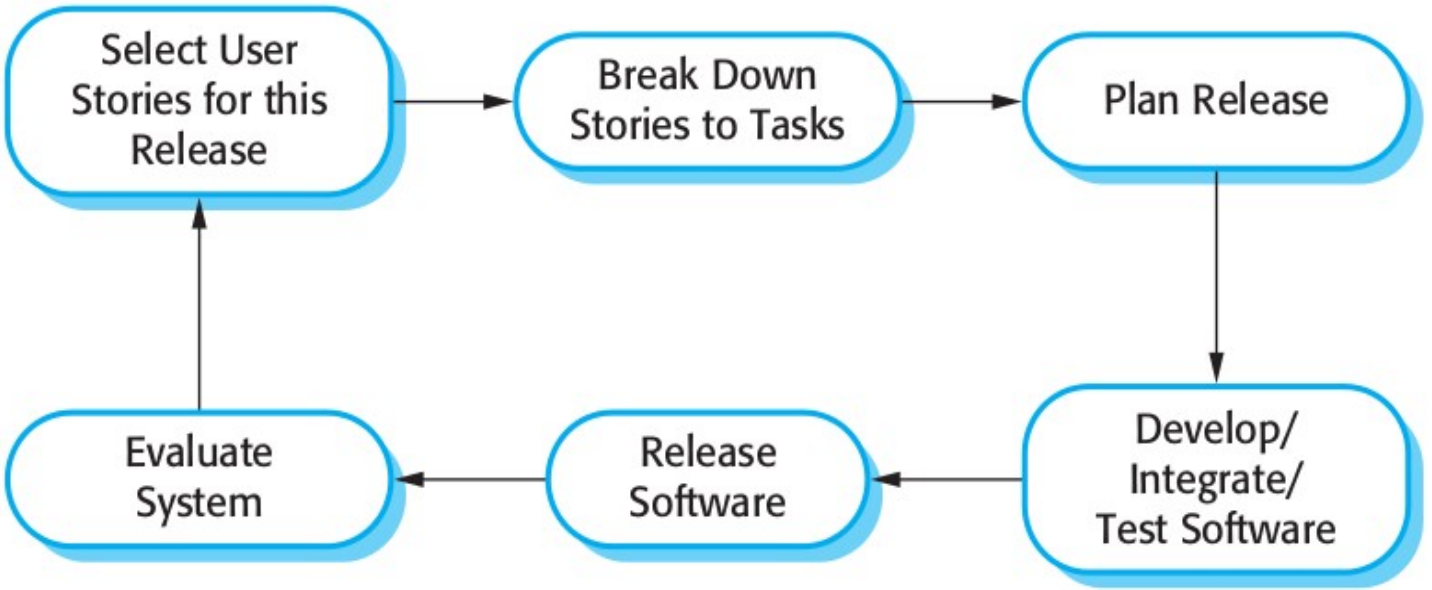
Plan-Güdümlü ve Çevik Geliştirim Dengesi

- Gerçekte çoğu yazılım projesi plan-güdümlü ve çevik yaklaşımlar içerir. Bunlar arasında bir denge sağlayabilmek için bir grup teknik, insani ve kurumsal soru cevaplanmalıdır:
  - Gerçekleştirmeye geçmeden önce detaylı bir tanımlama ve tasarım dokümanına ihtiyaç var mıdır?
  - Artımlı teslimat stratejisi gerçekçi mi?
  - Geliştirilen sistem ne kadar büyük?
  - Ne tür bir sistem geliştirilecek? Örn. gerçek zamanlı?
  - Sistem ömrü ne kadar olacak?
  - Sistem geliştirmeye destek teknolojiler kullanılabilir mi?
  - Takım düzeni nasıl?
  - Sistem geliştirimini etkileyen kültürel konular var mı?
  - Yazılım geliştirme takımındaki tasarımcı ve programcılar ne kadar iyi?
  - Sistem harici düzenlemelere tabi mi?

Ekstrem Programlama (XP)

- Çevik metotların en çok bilineni ve kullanılanıdır (Kent Beck, 2000)
- XP'de farklı programcılar tarafından geliştirilen sistem sürümleri bir günde entegre ve test edilebilir
- XP'de gereksinimler senaryolar (kullanıcı hikayeleri) şeklinde ifade edilir ve bunlar bir dizi görev olarak gerçekleştirilir
- Programcılar çiftler halinde çalışır ve kodu yazmadan önce testlerini yazarlar
- Yeni yazılan kodlar sisteme eklenmeden önce tüm testleri geçmiş olmalıdır
- Sistem sürümleri arasında kısa zaman aralıkları vardır

Ekstrem Programlama (XP)



Ekstrem Programlama (XP) İlkeleri

- **Küçük ve sık sürüm çıkarma** ile artımlı geliştirim desteklenmektedir. Gereksinimler basit kullanıcı hikayeleri veya senaryolarına dayanır.
- **Müşterinin sürece katılımı** geliştirme takımı ile müşterinin sürekli birlikte çalışması ile sağlanır. Müşteri tarafından bir temsilci geliştirim sürecinde yer alır ve kabul testlerini tanımlamadan sorumludur.
- **Süreç değil, insanlar anlayışı**, çiftler halinde programlama ve çok uzun çalışma saatleri olmayan sürdürülebilir bir geliştirme süreci ile desteklenir.
- **Değişimi kucaklama**, müşteriye düzenli olarak sistem sürümleri sunma, test-güdümlü geliştirim, refactoring ve yeni fonksiyonlarla sürekli entegrasyonu sağlamakla desteklenir.
- **Basitliği sağlama**, kod kalitesini sürekli olarak artırmak için refactoring ve basit tasarımlar kullanma.

Ekstrem Programlama (XP) Uygulamaları

- Artımlı planlama - Gereksinimler hikaye kartlarına kaydedilir. Geliştiriciler hikayeleri görevlere böler.
- Küçük sürümler – İş değeri olan fonksiyonların en küçük alt kümesi önce geliştirilir. Sistem sürümleri sık aralıklarla çıkarılır.
- Basit tasarım – Mevcut gereksinimleri karşılayacak en küçük tasarım yapılır.
- Önce testlerin yazıldığı geliştirim - Kodun kendisi geliştirilmeden önce gerekli testler yazılır.
- Refactoring – Tüm geliştiricilerin kodlarını sürekli refactor etmesi beklenir. Bu kodu basit ve kolay değiştirilebilir kılar.

Ekstrem Programlama (XP) Uygulamaları (devam)

- Çiftler halinde programlama – Geliştiriciler çiftler halinde çalışır, birbirlerinin yaptığı işi kontrol ederler ve iyi bir iş çıkması için birbirlerini desteklerler.
- Ortak sahiplenme – Çiftler sistemin her alanında çalışırlar, dolayısıyla sadece belirli insanların kodun herhangi bir bölümünde uzmanlaşmaları engellenmiş olur ve geliştiriciler tüm kodun sorumluluğunu taşırlar. Herhangi biri kodun herhangi bir yerinde değişiklik yapabilecek durumdadır.
- Sürekli entegrasyon – Bir görevle ilgili işler tamamlanınca tüm sisteme entegre edilir. Bu tip bir entegrasyondan sonra sistem tüm birim testlerini geçmelidir.
- Sürdürülebilir tempo – Çok miktarda fazla mesai kabul edilemez çünkü kod kalitesini düşürür ve orta vadede üretkenliği azaltır.
- İşyerindeki müşteri – Sistemin son kullanıcılarının bir temsilcisi XP takımının kullanımında olmalıdır. XP'de müşteri, geliştirim takımının bir parçasıdırve sistem gereksinimlerini geliştiricilere sağlamak zorundadır.

Hikaye Kartları

- Hikaye kartları, XP planlama sürecinin (planlama oyunu) ana girdileridir.
- XP sürecinde, müşteriler sistem gereksinimlerini oluşturmada ve öncelik sırasına sokmada yoğun görev alırlar.
- Gereksinimler gerekli sistem fonksiyonları listesi şeklinde oluşturulmaz. Müşteri, geliştirme ekibi ile senaryoları tartışır.
- Hepbirlikte müşteri ihtiyaçlarını kapsayan hikaye kartlarını oluştururlar. Geliştirme takımı da bir sonraki sürüme bu senaryoyu ekler.
- Hikaye kartları oluşturulduktan sonra geliştiriciler bu hikayeleri görevlere bölerler ve her görevi tamamlamak için gereken kaynak ve zaman tahmininde bulunurlar.

Hikaye Kartları

- Planlama oyunu esnasında bazen gereksinimler net olarak belirlenemez, sorulara net cevaplar verilemez.
- Bu durumda geliştirme takımı bir prototip veya deneme sürümü ortaya çıkarabilir. Bu adıma “spike” adı verilir ve XP jargonunda programlama yapılmayan süreç anlamını taşır. Bazen “spike” adımlarda sistem mimarisi geliştirilir veya dokümantasyon yapılır.
- XP, geleneksel yöntemlerin aksine değişime hazır tasarım yapma işini temel olarak almaz. Değişikliğin kesinlikle gerçekleşeceğini, dolayısıyla değişiklik gerçekleştiğinde sistemi yeniden organize etmeyi kabul eder.

Hikaye Kartı Örneği

- Reçete yazma hikayesi

  Ayşe kliniğine gelen bir hastaya reçete yazmayı istemektedir. Hasta kaydı ekranında gözükmektedir ve Ayşe burada ilaç sahasına tıklayıp “mevcut ilaç”, “eski ilaç” ve “ilaç listesi” seçeneklerinden birini seçebilir.

  Eğer “mevcut ilaç” seçeneğini seçerse sistem ondan dozunu kontrol etmesini ister. Eğer dozu değiştirmek isterse yeni dozu girer ve reçeteyi onaylar.

  Eğer “yeni ilaç” seçeneğini seçerse sistem hangi ilacı yazacağını bildiğini varsayar. İlaç adının ilk birkaç harfini girer. Sistem de ona o harflerle başlayan ilaçların listesini görüntüler. Gereken ilacı seçer ve sistem ona seçtiği ilacın doğru olup olmadığını sorar. Dozu girer ve reçeteyi onaylar.

  Eğer “ilaç listesi” seçeneğini seçerse sistem onaylanmış ilaçlar için bir arama kutusu gösterir. Böylece gerekli ilacı arayabilir. Bir ilaç seçer ve doğru olup olmadığı ona sorulur. Dozu girer ve reçeteyi onaylar.

  Sistem her zaman dozun onaylı bir aralıkta olup olmadığını kontrol eder. Eğer değilse Ayşe'den dozu değiştirmesi istenir.

  Ayşe reçeteyi onayladıktan sonra kontrol için reçete görüntülenir. 'Tamam' veya “Değiştir”e tıklar. Eğer “Tamam”ı tıklarsa reçete veritabanına kaydedilir. Eğer “Değiştir”i tıklarsa “Reçete yazma” sürecine yeniden girer.

Hikayeyi Görevlere Ayırma

- Reçetelenmiş ilacın dozunu değiştirme
- İlaç seçimi
- Doz kontrolü
  - Doz kontrolü doktorun tehlike oluşturacak şekilde az veya çok doz reçete etmesini önlemek için bir önlemdir.
  - İlaç ID kullanılarak listeden tavsiye edilen en küçük ve en büyük doz bilgisine ulaşılır.
  - Reçete edilen dozun gerekli aralıkta olup olmadığı kontrol edilir. Aralığın dışında ise bir mesaj ile dozun çok küçük veya çok büyük olduğu belirtilir. Eğer aralık içindeyse “Onay” düğmesi aktif hale getirilir.

XP'de Test

- Artımlı geliştirim ve plan-güdümlü geliştirim arasındaki temel fark sistemin test edilme yöntemleridir.
- Bazı artımlı geliştirim yaklaşımlarında test ekibinin kullanabileceği bir sistem tanım dokümanı bulunmadığından plan-güdümlü geliştirmeye göre testler oldukça gayrı resmi yapılır.
- XP, programın test edilmesinin önemine vurgu yapar.

XP'de Testin Temel Özellikleri

- Önce-test yöntemi ile geliştirim
- Senaryolardan artımlı test geliştirimi
- Test geliştirimi ve doğrulamada kullanıcının da yer alması
- Otomatik test çerçevelerinin kullanılması

Çiftler Halinde Programlama (Pair Programming)

- Çiftler aynı bilgisayarda kod geliştirirler
- Dinamik bir eşleştirme kullanarak, tüm takım elemanlarının birlikte kod geliştirmesi sağlanır
- Yararları
  - Takımın tamamı kodun tamamı üzerinde bilgi sahibi olur
  - Kod iyi bir şekilde kontrol edilmiş olur
  - Refactoring desteklenmiş olur
  - Belirli bir zamanda iki ayrı kişinin yazdığı kodun yarısı kadar daha fazla kod geliştirilir

Çevik Proje Yönetimi

- Yazılımın zamanında ve planlanan bütçe dahilinde teslim edilmesi proje yöneticisinin temel sorumluluğudur
- Proje yöneticileri yazılım mühendislerinin yaptıkları işi denetlerler ve yazılım geliştirmenin nasıl ilerlediğini gözlemlerler
- Proje yönetimine standart yaklaşım plan-güdümlüdür. Bu yaklaşım, ne teslim edilecek, ne zaman teslim edilecek, kimler çalışacak, gibi konuların net olarak belirlenmesini gerektirir
- Bu yaklaşım çevik yöntemlerle gerçekleştirilemez ancak çevik projelerin de yönetilmeye ihtiyacı vardır
- Kaynaklar ve zamanın takımın kullanımına sunulması ile yeni bir yönetim anlayışı ortaya çıkar ve bu anlayış artımlı geliştirme ve bazı çevik yazılım geliştirme prensiplerini bünyesinde birleştirir

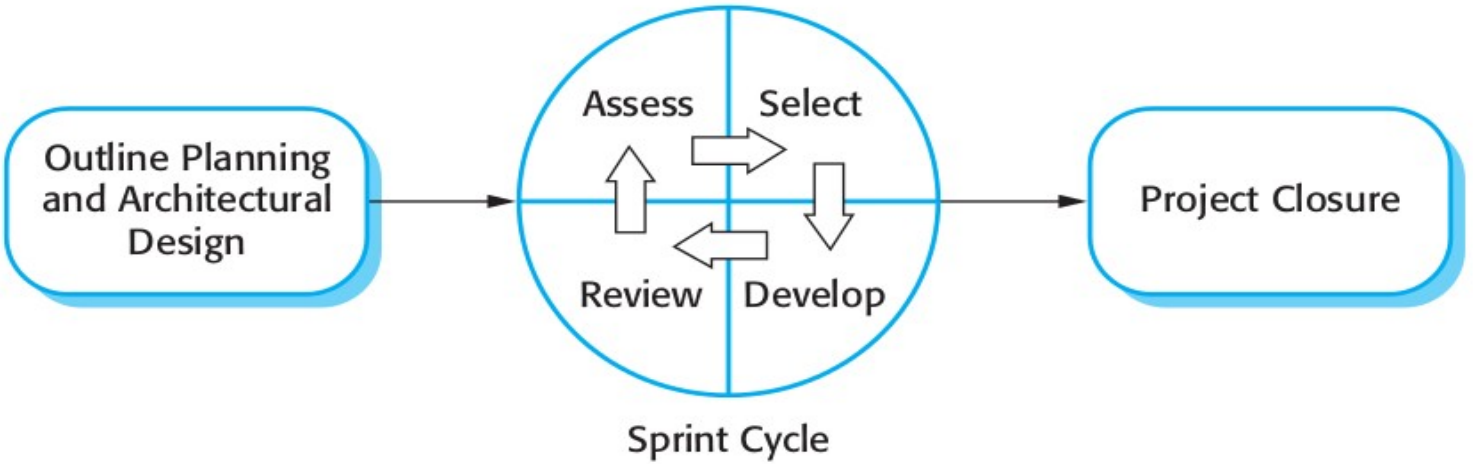
Scrum

- Schwaber, 2004; Schwaber ve Beedle, 2001
- **Scrum** genel bir çevik yöntemdir ancak çevik geliştirime getirilen yeni teknik yaklaşımdan ziyade artımlı geliştirimin yönetimine odaklanır
- Çiftler halinde programlama ve önce-test yöntemi gibi yöntemler önermez, dolayısıyla XP gibi, projeye yönetim çatısı sağlamak gibi daha teknik çevik yaklaşımlarla kullanılabilir

Scrum Süreci

- Scrum'da üç faz bulunur
  - Taslak planlama - projenin genel amaçları belirlenir ve tasarımı yapılır
  - Hızlı koşma döngüsü – her döngüde sistemin bir artışı gerçekleştirilir
  - Proje kapatma – proje toparlanır, gerekli dokümanlar (yardım dokümanı, kullanma kılavuzu, vb.) tamamlanır, projeden öğrenilen dersler değerlendirilir

Scrum Süreci



Scrum Süreci

- Scrum'ın yenilikçi tarafı merkezdeki sürecidir . Hızlı koşma döngüsü, hangi işin yapılacağının değerlendirmesi, geliştirilecek özelliklerin belirlenmesi ve yazılımın gerçekleştirimi gibi süreçler içeren bir planlama birimidir. Döngü sonunda tamamlanmış fonksiyonellik müşteriye sunulur.
- Bu sürecin anahtar noktaları şunlardır:
  - Koşu döngüleri 2-4 haftalık sabit uzunluğa sahiptirler. XP'deki sürüm çıkarma işine denk gelir.
  - Projede yapılacaklar listesi üzerinden planlama başlar. Değerlendirme sürecinde bu liste gösden geçirilir, öncelikler ve riskler tanımlanır. Müşteri bu sürece dahil edilir.
  - Müşteri ile çalışan tüm proje takımı o koşu boyunca geliştirilecek işlerin seçimi için katkıda bulunur.
- Hemfikir olunduğunda takım yazılımı geliştirmek için organize olur. Kısa, günlük toplantılar yapılır. Yazılım geliştirme ekibi bu sürecte müşteriden ve yönetimden izole edilir. Tüm iletişim ‘Scrum master’ üzerinden devam eder.
- Koşu sonunda, yapılan iş kontrol edilir ve müşteriye sunulur. Sonra da bir sonraki koşu döngüsü başlar

Scrum'ın başarısı

- Scrum'ın başarılı olduğu bir telekomünikasyon projesinde kaydedilen avantajları
  - Ürün yönetilebilir ve anlaşılabilir parçalar ayrılmıştır
  - Değişen gereksinimler ilerlemenin önünde engel değildir
  - Takım iletişimi artmıştır ve herşey tüm takıma görünürdür
  - Müşteriler zamanında ve artımlı teslimat sayesinde geri bildirim verebilmektedir
  - Müşteri ve geliştiriciler arasında güven kurulur, projenin başarılı olacağına dair olumlu bir atmosfer ve kültür oluşur

Çevik Yöntemlerde Ölçeklenebilirlik

- Küçük takımlar, küçük veya orta ölçekli projeler
- Hızlı teslimata büyük yazılımlar için de ihtiyaç duyulabilir
- Büyük sistem geliştirme, küçük sistem geliştirmeden birçok yönden ayrılır:
  - Büyük sistemler genelde birbiri ile iletişim kuran ayrı sistemler bütünüdür, takımlar ayrıdır
  - Kod geliştirme kadar sistemlerin birlikte çalışması için konfigürasyonlar iş yükü oluşturur
  - Büyük sistemler, kanunlar ve düzenlemelerden etkilenir
  - Uzun geliştirim süreleri var, aynı takımı korumak mümkün değil
  - Çok fazla çeşitte paydaş var

Çevik Yöntemlerde Ölçeklenebilirlik

- Çevik yöntemleri büyük ölçekli şirketlerde kullanmak birkaç sebepten dolayı zordur:
  - Çevik yöntemlere yabancı olan proje yöneticileri risk almak istemezler, yeni yaklaşımın ürünlerini nasıl etkileyeceğini kestiremezler.
  - Büyük kuruluşlar kalite prosedürlerine ve standartlarına sahiptir. Bürokratik sebeplerden değişime sıcak bakılmaz.
  - Takım elemanlarının kişisel becerilerinin yüksek olduğu durumlarda çevik yöntemler daha iyi çalışır. Büyük kuruluşlarda farklı niteliklerde çok sayıda insan bulunduğundan düşük yetenekli kişiler takımda etkin olamayabilirler.
  - Özellikle uzun yıllardır geleneksel sistem mühendisliği süreçlerini kullanan kurumlarda çevik yöntemlere karşı kültürel bir direniş de oluşmaktadır.

Önemli Noktalar

- Çevik yöntemler hızlı geliştirime, sık yazılım sürümü çıkarmaya, sürecin ek yüklerini azaltmaya ve yüksek kalitede kod üretmeye odaklanan artımlı geliştirim yöntemleridir. Müşteriyi doğrudan sürece dahil ederler.
- Çevik veya plan-güdümlü yaklaşım kullanmanın seçimi üretilen yazılımın tipine, geliştirme takımının yeteneklerine ve yazılımı geliştiren firmanın kültürel yapısına bağlı olmalıdır.
- Ekstrem Programlama (XP) bünyesinde hızlı sürüm çıkarma, müşteriyi geliştirme sürecine dahil etme, sürekli yazılım iyileştirme gibi birçok iyi programlama uygulamasını barındıran, iyi bilinen bir çevik yöntemdir.
- XP'nin güçlü yanlarından biri geliştirimden önce testlerin oluşturulmasıdır.
- Scrum yöntemi proje yönetim framework sunan bir çevik yöntemdir. Merkezinde birçok sistem artımının yapıldığı sabit zaman aralığı olan “sprint” bulunur.
- Çevik yöntemleri geniş sistemlere uygulamak zordur. Geniş sistemler belli bir tasarım ve dokümantasyona ihtiyaç duyarlar. Sürekli entegrasyon farklı yerlerde bulunan takımlarla imkansız hale gelmektedir.

Haftaya görüşmek üzere