YMT312 Yazılım Tasarım ve Mimarisi

Birleşik Süreç ve Çevik (Agile) Yazılım Süreç Modelleri

Bölüm-3

Doç. Dr. Resul DAŞ

Fırat Üniversitesi Yazılım Mühendisliği Bölümü

Bu Haftaki Konular

Amaçlar

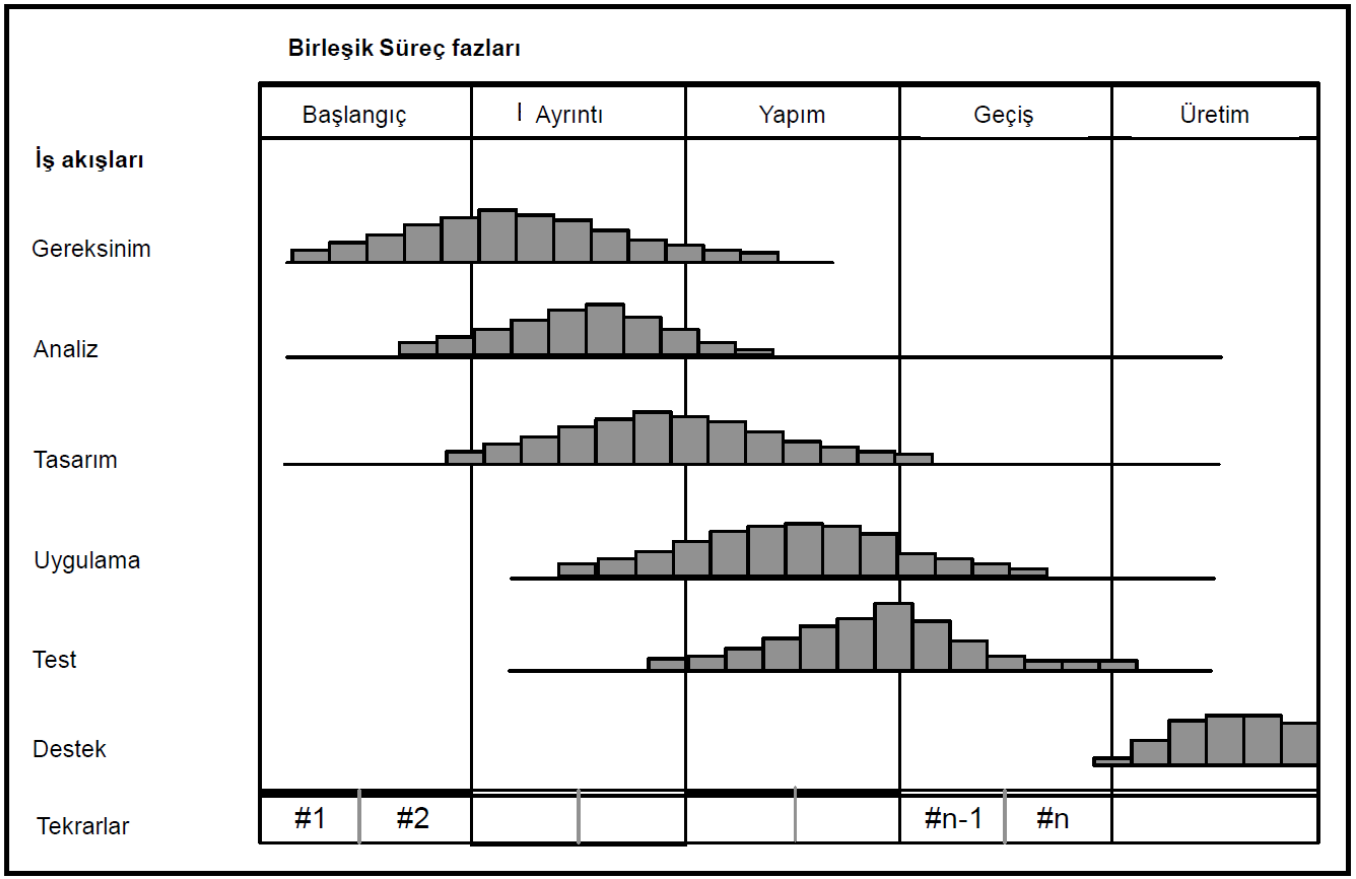
- Birleşik Süreç ve Fazlarını kavramak
- Çevik (Agile) yaklaşım.
- Çevik (Agile) Yazılım Süreç Modellerini Anlamak
- Çevik (Agile) Yazılım Süreçlerinin diğer süreçlere göre üstünlükleri.
- Çevik (Agile) Yazılım Süreç Modellerinin kullanım amaçları.
- Yazılım Süreç Modelleri seçimi.
- Yazılım Süreçleri – IEEE/IEA 12207 kavramı.

Birleşik Süreç (Unified Process)

- Nesneye dayalı yazılım geliştirmek için var olan yöntemlerin deneyimler sonucu kabul gören en iyi özellikleri bir araya getirilerek tümleştirilmiş yazılım geliştirme süreci (The Unified Process - UP) oluşturulmuştur.

- **Yinelemeli (iterative):** Problemdeki istekler (requirements) bir bütün olarak yerine getirilmeye çalışılmaz. Önce problemin bir kısmı ele alınır. Problemin bu kısmı bağımsız bir proje olarak ele alınır ve hedeflenen istekleri yerine getiren sınanmış tam bir ürün ortaya çıkartılır. Ardından bir sonraki yinelemeye geçilir ve yeni istekler ele alınır. Her iterasyon sonunda hedefe daha yakın bir ürün elde edilir.
- **Arttırmalı ve evrimsel (incremental, evolutionary):** Her iterasyon adımında yeni istekler ele aldığında iterasyonlar sonucunda elde edilen ürünlerin özellikleri artar ve hedeflenen yazılım ürününe yaklaşırlar.
- **Risk güdümlü (Risk-driven):** ilk iterasyonlarda en riskli kısımlar gerçeklenmelidir. Böylece daha projenin ilk aşamalarında ortaya çıkabilecek problemler görülebilir ve gerekli önlemler alınabile. Örnegin zaman planı gözden geçirilir, ekibe yeni elamanlar alınabilir, bütçe güncellenebilir.

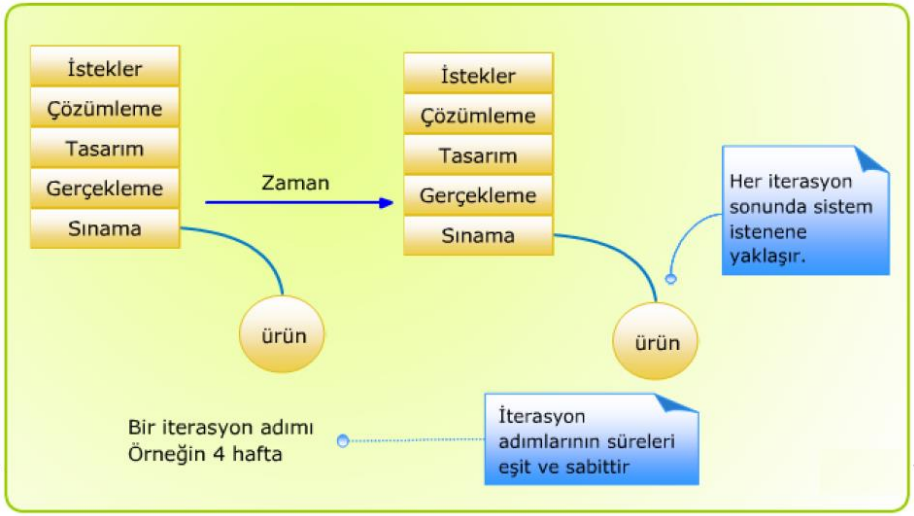
Birleşik Süreç Fazları



UP: Yinelemeli ve evrimsel yazılım geliştirme

Her yineleme (iterasyon) adımında bütün bir yazılım projesi varmış gibi davranılır.

Gerekli tüm aşamalardan geçilerek sınanmış, çalışır bir ürün elde edilir.



Yinelemeli Sürecin Yararları

- Değişen isteklere uyum: Yazılım geliştirmedeki en büyük problemlerden biri isteklerin değişmesidir. Yinelemeli çalışma sayesinde istekler değişirse bir sonraki yineleme adımında değişikliklere uyum sağlanabilir.
- Erken geri besleme: Her iterasyondan sonra müşteriden (yazılımın kullanıcısı) geri besleme alınarak beklentilerin ne ölçüde karşılandığı sınanmış olur. Eğer bir uyumsuzluk varsa erkenden önlem alınır.
- Büyük sistemlerde çözümleme kolaylığı: Büyük ve karmaşık bir sistemi bir bütün olarak tek adımda çözümlemek (anlamak) zor olacaktır. Bu nedenle istekleri parça parça ele almak sistemi anlamak ve tasarlamak açısından kolaylık sağlar.
- Her iterasyonda deneyim kazanılması: Her yineleme adında yazılım ekibi o konu ile ilgili deneyim kazanır ve elemanlar birbirini daha iyi tanır.
- Risklerin erken giderilmesi (eğer mümkünse): Riskli kısımlar ilk iterasyonlarda ele alındığından eğer bir problem varsa bu problemlere karşı önlemler mümkün olduğu kadarıyla erken alınmış olur.
- Erken ürün elde etme, takımda moral yükselmesi: Her iterasyon sonunda bir ürün elde edildiğinden zaman içinde hedefe yaklaşıldığı görülür ve yazılım ekibi çalışmalarının sonuçlarını görerek moral kazanır.

UP'nin Kullanılmasına Yönelik Öneriler

2 - 6 haftalık sabit süreli iterasyonlar uygulanmalı

- Deneyemlere göre bir iterasyonun süresinin 3 ya da 4 hafta olarak seçilmesi iyi sonuçlar vermektedir.
- İterasyon 2 haftadan kısa olursa bu sürede bir ürün çıkarmak mümkün olmaz.
- Altı haftadan daha uzun süreli iterasyonlarda ise erken geri besleme alma olanağı ortadan kalkar ve çok fazla sayıda istek ile uğraşmak gerekir.

Yüksek risk taşıyan kısımlar ilk iterasyonlarda gerçeklenmeli

- Daha önce de açıklandığı gibi ortaya çıkabilecek problemleri mümkün olduğu kadar erken fark edip bunlara karşı önlem alabilmek için zorlu görünen istekler önce ele alınmalı.

Temel oluşturan yapılar (çekirdek) önce gerçeklenmeli

- Yüksek riskli kısımlardan başka önce ele alınması gereken modüller sistemin temelini (iskeletini) oluşturan yapılardır.

UP'nin Kullanılmasına Yönelik Öneriler

- Sürekli kullanıcılardan geri besleme alınmalı, isteklere uyulmaya dikkat edilmeli
- Her iterasyondan sonra ürün tam olarak sınanmalı
  - Her iterasyonda bir ürün oluşturulduğu unutulmamalı ve bu ürün tam olarak sınanmalıdır.
    - Aksi durumda hatalar geç fark edilir ve bunları düzeltme maliyeti yüksek olur.
- Kullanım senaryoları yöntemi (use case) uygulanmalı
- Görsel modelleme (UML) kullanılmalı
  - UML tasarımların ifade edilmesini ve takım elemanları arasında iletişimi kolaylaştırır.
- Bir iterasyonda elde edilen deneyim diğer iterasyonda kullanılmalı

Çevik (Agile) Yaklaşım

Bir yazılım projesi, 6 ay içerisinde tamamlanmazsa ve projeye müşterinizi dahil etmezseniz, başarıya ulaşma ihtimaliniz zayıftır.

Çevik (Agile) Yaklaşım

Çevik (Agile) Modelleme

- Scott Ambler tarafından önerilmiştir.

- Çevik model prensiplerinin bir kümesini önerir.
  - Bir amaç ile modelle.
  - Çoklu model kullan.
  - Travel light.
  - İçerik gösterimden daha önemlidir.
  - İçeriği oluşturmada kullandığın model ve araçları bilmek.
  - Adapt locally

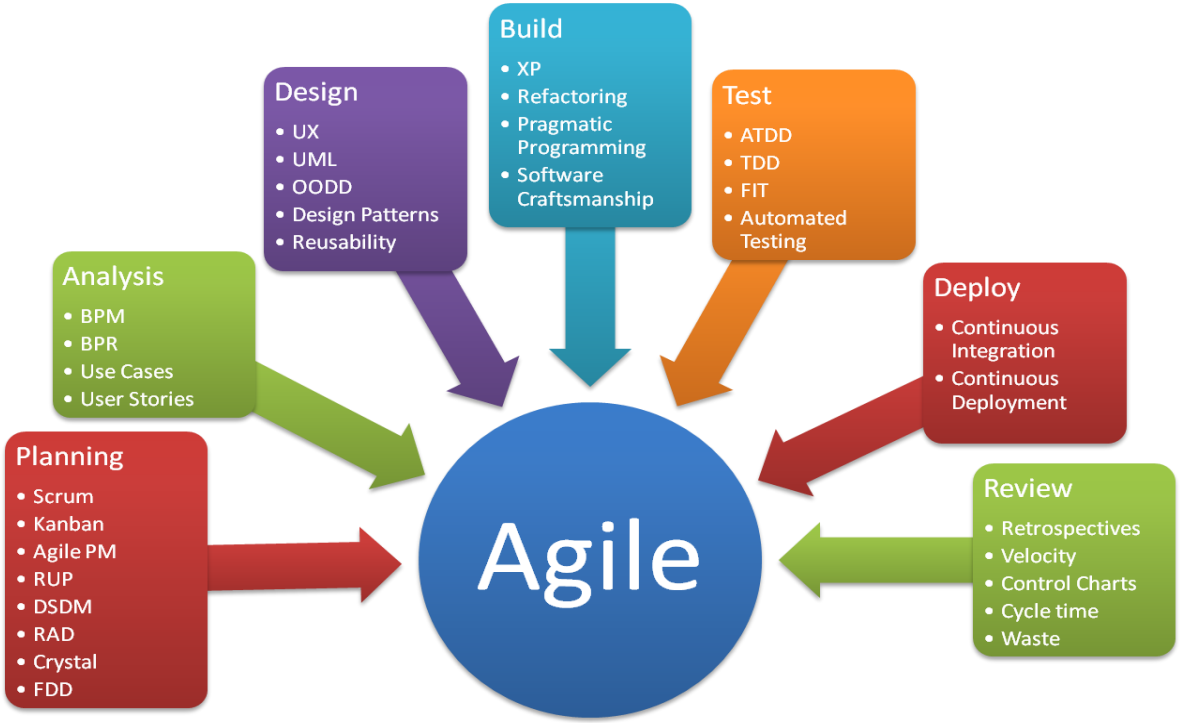
Çevik (Agile) Yazılım Süreç Modelleri

- Çevik modeller, mevcut geleneksel modellere alternatif olarak, 1990'larda ortaya çıkmaya başlamıştır.
  - **Çevik:** Kolaylık ve çabuklukla davranan, tetik, atik. [TDK]
  - 1950'lerdeki üretim alanında verimliliğin artırılması için geliştirilen yalın yaklaşımların, yazılım sektöründe bir uzantısı olarak ortaya çıkmıştır.

- Çevik modeller kapsamında; yazılım sistemlerini etkili ve verimli bir şekilde modellemeye ve belgelendirmeye yönelik, pratiğe dayalı yöntemler yer alır.

- Bu modelleme biçiminin; kapsadığı değerler, prensipler ve pratikler sayesinde geleneksel modellemelere metotlarına göre yazılımlara daha esnek ve kullanışlı biçimde uygulanabileceği savunulmaktadır.

Çevik (Agile) Yazılım Süreç Modelleri



Çevik Yazılım Geliştirme Manifestosu

- 2001 yılında, dünyanın önde gelen çevik modellerinin temsilcileri, ortak bir zeminde buluşabilmek adına bir araya gelerek "çevik yazılım geliştirme manifestosu" nu yayınlamışlardır. Bu manifestoya göre;

  - Bireyler ve aralarındaki etkileşim, kullanılan araç ve süreçlerden;
  - Çalışan yazılım, detaylı belgelerden;
  - Müşteri ile işbirliği, sözleşmedeki kesin kurallardan;
  - Değişikliklere uyum sağlayabilmek, mevcut planı takip etmekten;

  daha önemli ve önceliklidir.

  (Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas)

Çevik Yazılım Geliştirme Manifestosu- Prensipler

- En önemli önceliğimiz müşteriyi değerli yazılımı erken zamanda ve sürekli bir şekilde sunarak memnun etmektir.

- Geliştirme sürecinde son zamanlarda da gelse değişen gereksinimler hoş karşılanmalıdır. Çevik süreçler değişimi müşterinin rekabetçi avantajı için kullanırlar.

- Çalışan yazılım bir iki haftadan bir iki aya kadar, mümkün olan en kısa sürelerde sunulmalıdır.

- Geliştiriciler ve iş insanları proje boyunca beraber çalışmalıdırlar.

Çevik Yazılım Geliştirme Manifestosu- Prensipler

- Projeleri motivasyonu yüksek bireylerin çevresinde geliştirilmelidir. Onlara ihtiyaçları olan destek ve ortamı sağlayın ve işi yapabileceklerine güvenin.

- Bilginin takımlar içerisinde ya da takımlar arasında en iyi paylaşılması ve anlaşılmasının yolu yüz yüze iletişimdir.

- İlerlemenin en iyi göstergesi çalışan yazılımdır.

- Çevik süreçler süründürülebilir geliştirme sağlar. Sponsorlar, geliştiriciler ve kullanıcılar sabit hızlarını sonsuz şekilde koruyabilirler

Çevik Yazılım Geliştirme Manifestosu- Prensipler

- Teknik mükemmellik ve iyi tasarıma verilen sürekli önem çevikliği arttırır.
- Basitlik önemlidir. (Yapılmaması gereken tüm işleri önlemek).
- En iyi mimari, gereksinimler, ve tasarımlar kendi kendilerine organize olabilen takımlardan çıkar.
- Düzenli aralıklarla, takımlar nasıl daha etkili olacaklarına bakar, davranışlarını düzenler ve ona göre hareket eder.

Çevik Modellemenin Başlıca Özelliği

- Veri modelleri ve ara yüzü modelleri gibi modelleme tekniklerinin neler olduğunu ve bunların ayrıntılarını söylemek yerine bu tekniklerin nasıl uygulanması gerektiğini söylemesidir.

- Mesela; yapılan projelerin test edilmesi gerektiğini belirtse bile nasıl test hazırlanacağına değinmemesi gibi.

- Bu sadece bir yöntemler biçimidir ve bir projenin etkili, hızlı bir şekilde ortaya çıkarılmasında, müşterinin ihtiyaçlarını karşılamasında ve aynı zamanda da her türlü değişikliğe kolayca adapte olabilmesinde geliştiricilere yol gösterir.

Hangi Durumlarda Kullanılabilir?

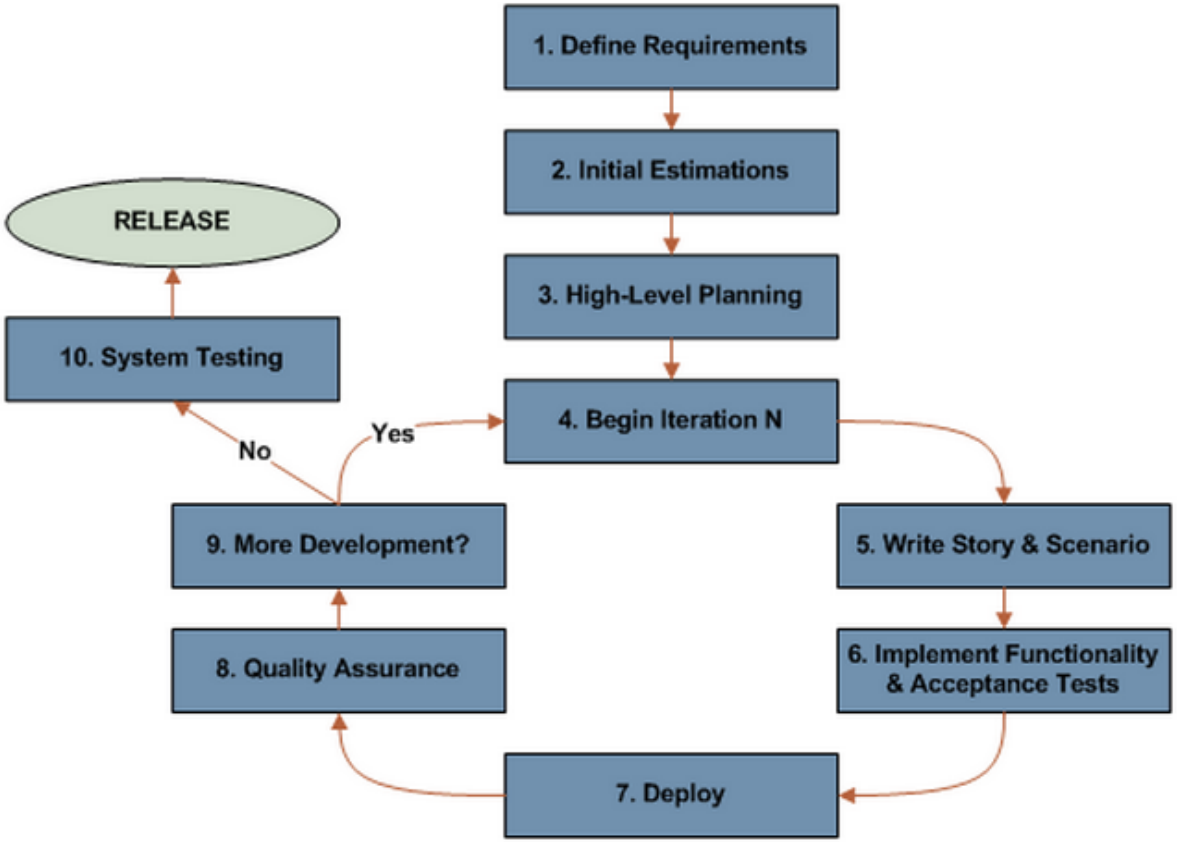
Bu metotların kullanılmasının en uygun olduğu durumlar şunlardır:

- Projenin yazılım evresinde müşteriden gelebilecek talep değişikliklerinin tahmin edilemez olması.
- Projenin parçalarının önce tasarlanıp ardından hemen geliştirilmesinin gerekmesi ve önceden ne yapılacağını, detaylı yol haritasını ve tasarımını tahmin etmenin çok güç olması.
- Analiz, tasarım ve test etme süreçlerinin ne kadar zaman alacağının önceden bilinememesi.
- Yazılım ekibinin birlikte çalışmak, hiyerarşiye önem vermemek, sağlam iletişim kurmak gibi özelliklere sahip olması.

Çözüm


Müşterini projeye dahil et
En önemli özellikleri ilk önce yap
Mümkün olan en kısa sürede çalışan bir sürüm çıkar
Müşteri ile değerlendir
Geri bildirimlere göre yeniden düzenle

Çevik Yazılım Geliştirme Süreci



Çevik Yazılım Modelinin Diğer Modellerden Farkı

- Çevik modeller bazen planlı ve disiplinli olmaması ile eleştirilse de bu doğru değildir.

- Açıklamak gerekirse; bir projenin ortaya çıkarılmasında 2 tip model kullanılabilir. Biri uyarlanabilir olan, bir diğeri de tahmin edilebilir olandır.

- Değişime tepki verebilecek şekilde tasarlanmıştır. Mesela, projenin gereksinimleri değişirse, projenin takımı da bu değişikliğe ayak uydurur ve değişir.

- Bu takım bize 1 hafta sonra hangi işlerin yapılacağını söyleyebilir ancak 1 ay sonra ne yapacaklarını belirtemez.

Çevik Yazılım Modelinin Diğer Modellerden Farkı

- Yani projede planlanan tarih ne kadar uzaksa o tarihte yapılacak işler de bir o kadar belirsizdir.
- Tahmin edilebilir olan modellerin takımları projenin tamamlanma süresi içindeki tüm değişikliklerin ve gelişmelerin tarihini önceden bilir, bu plan çerçevesinde iş yapar.
- Değişiklik olduğunda tüm plan iptal olur ve yeni bir program yapılır.
- Yeni programda ise sadece en önemli olan değişikler seçilip projeye dahil edilir.
- Çevik modeller uyarlanabilir olanlara dahildir.
- Yani değişime açıktır ve uyarlanabilirdir.
- Uyarlanabilir olduğu için bu kadar net bir şekilde planlanmayan proje müşterinin isteklerine öre şekil alır ve istenilen başarıyı gösterir.

Çevik Yazılım Modelinin Diğer Modellerden Farkı

Çevik modellerin bir diğer özelliği de;

- Diğer yinelemeli geliştirme modellerin (iterative development models) zaman dilimi ay olarak alınırken, çevik modellerde bu süre haftaya kadar düşer.

- Bu kısa süre içinde küçük ilerlemeler şeklinde ve her adımda müşteriden geri bildirim alınarak yazılım ortaya çıkarılır.

Çevik Yazılım Geliştirme Modelleri

Çevik modeller tam bir yazılım süreci değildir ama kapsamlı yazılım geliştirme yöntemlerini tamamlayıcı niteliktedir.

Başlıca çevik yazılım geliştirme süreç modelleri:

- Uçdeger Programlama (“Extreme Programming – XP”)
- Adaptif Yazılım Geliştirme (”Adaptive Software Development -ASD”)
- Dinamik Sistem Geliştirme Metodu (”Dynamic System Development Method”)
- SCRUM
- CRYSTAL
- Özellik Güdümlü Gelistirme (“Feature-Driven Development – FDD”)
- Çevik Tümlesik Süreç (“Agile Unified Process – AUP”)
- Çevik bilgi Metodu (Agile Data Method)

Uçdeğer Programlama (Extreme Programming – XP)

- Uçdeğer programlama, Kent Beck tarafından 1999 yılında bir yazılım geliştirme disiplini olarak ortaya çıkarılmıştır.
- Tüm gereksinimler senaryolar şeklinde oluşturulur. Daha sonra senaryolar işlere bölünür.
- Yazılımcılar çiftler halinde çalışır ve her iş için test de geliştirir. İşleri sonra tümleştirir.
- Sistemin müşterisi de geliştirici takımın devamlı bir parçasıdır.

4 prensip etrafında toplanır:

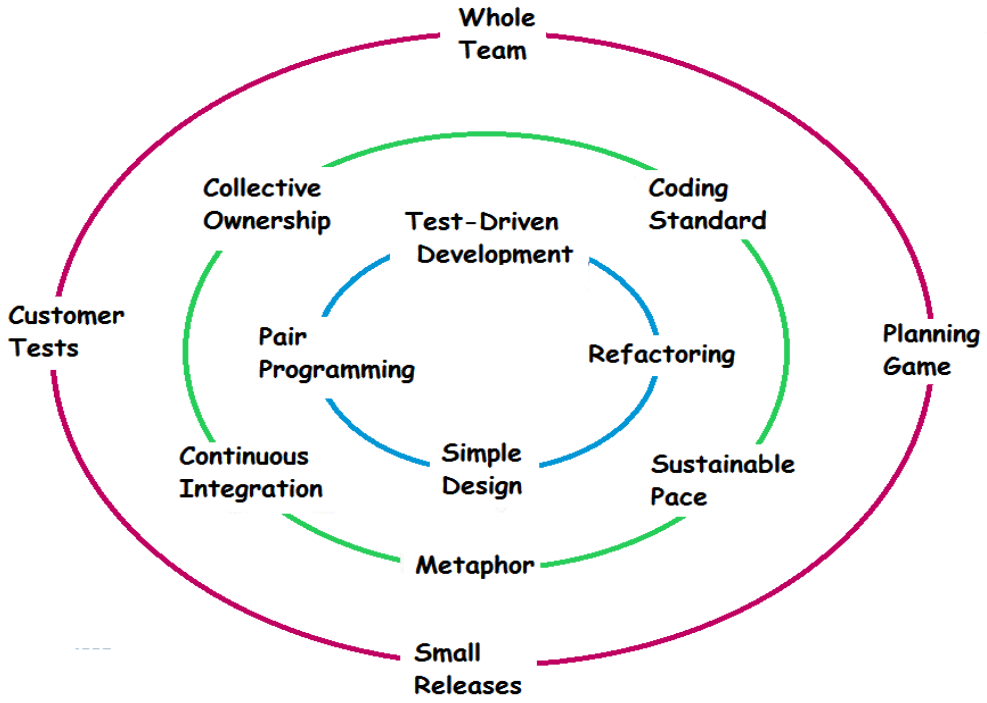
- Basitlik,
- İletişim,
- Geri bildirim,
- Cesaret

Uçdeğer Programlama

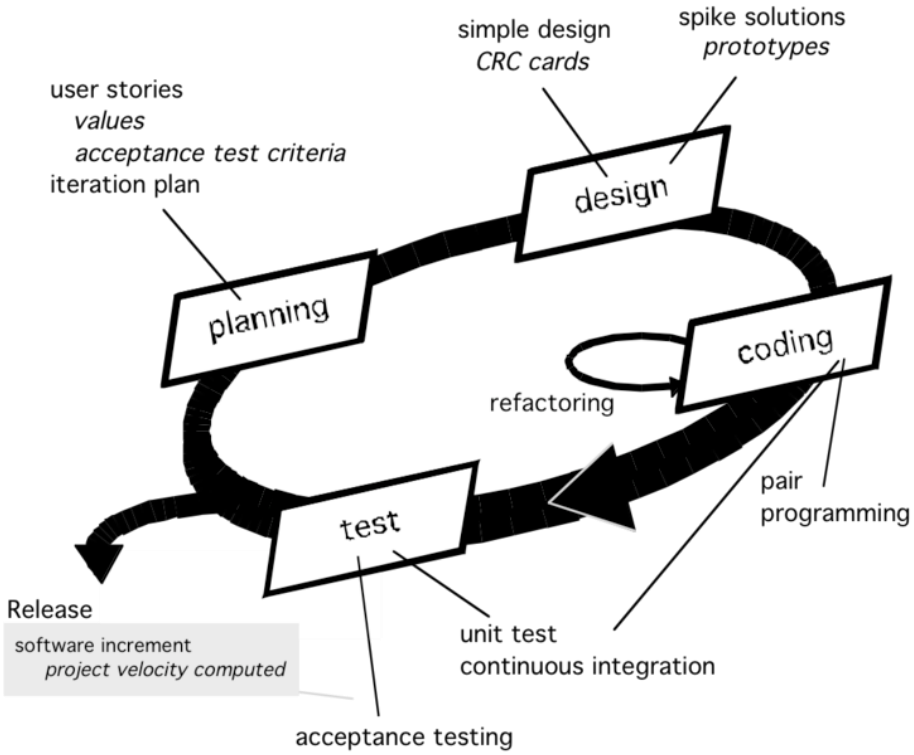
12 temel pratiğin birlikte uygulanmasını gerektirir:

- Planlama oyunu
- Kısa aralıklı sürümler
- Müşteri katılımı
- Yeniden yapılandırma (“refactoring”)
- Önce test (“test first”)
- Ortak kod sahiplenme
- Basit tasarım
- Metafor
- Eşli programlama (“pair programming”)
- Kodlama standardı
- Sürekli entegrasyon (“continuous integration”)
- Haftada 40 saat çalışma

Uçdeğer Programlama



Uçdeğer Programlama



Uçdeğer Programlama

Bazı özellikleri

- Takımın bilgisayarları kübiklerle bölünmüş büyük bir odanın ortasında yer alır.
- Bir müşteri temsilcisi geliştirici takımlarla devamlı beraber çalışır.
- Hiç kimse aynı iş için peşpeşe iki haftadan fazla çalışamaz.
- Takım içerisinde özelleşme yoktur. Herkes belirtim, tasarım, kodlama ve test aşamalarında görev yapar.
- Parçalar geliştirilmeden önce ayrı büyük bir tasarım aşaması yoktur. Onun yerine parçalar geliştirilirken tasarım da değiştirilir.

Küçük ve orta ölçekli projelerde kullanılırlar

Kullanıcın gereksinimleri belirsiz ya da değişkense kullanışlıdırlar.

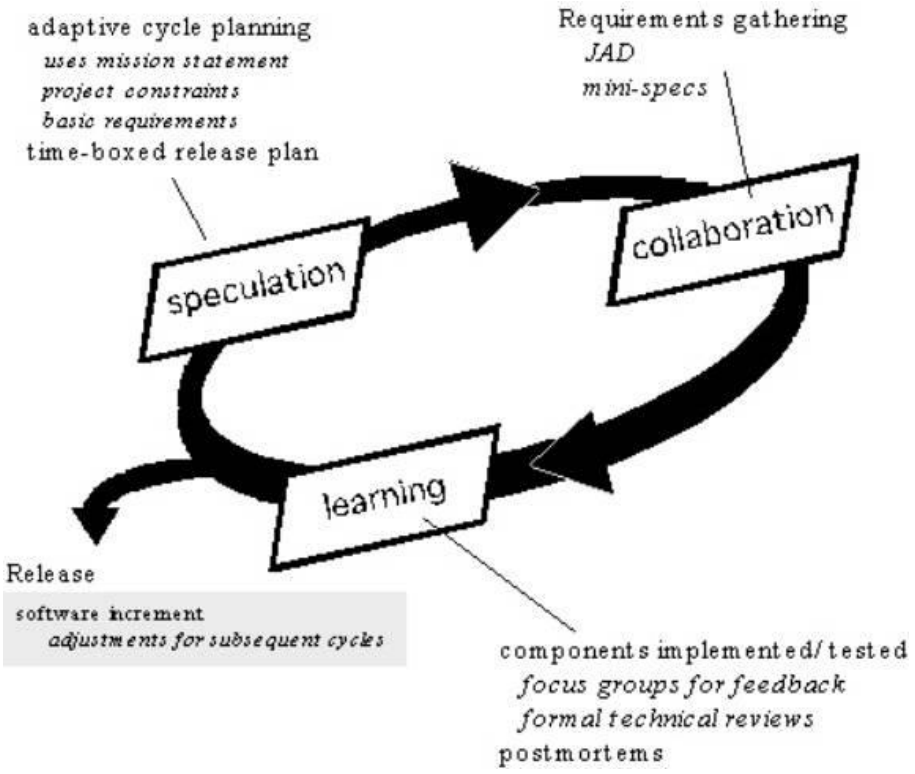
Adaptif Yazılım Geliştirme (Adaptive Software Development -ASD)

ASJim Highsmith tarafından önerilmiştir

D — ayırt edici özellikleri

- Göreve-dayalı planlama.
- Bileşene-dayalı odak
- “zaman aralıkları” kullanır
- Riskleri açık şekilde göz önünde bulundurur
- Gereksinim toplamada işbirliğini vurgular
- Süreç boyunca “öğrenme” yi vurgular

Adaptif Yazılım Geliştirme



Dinamik Sistem Geliştirme (”Dynamic System Development”)

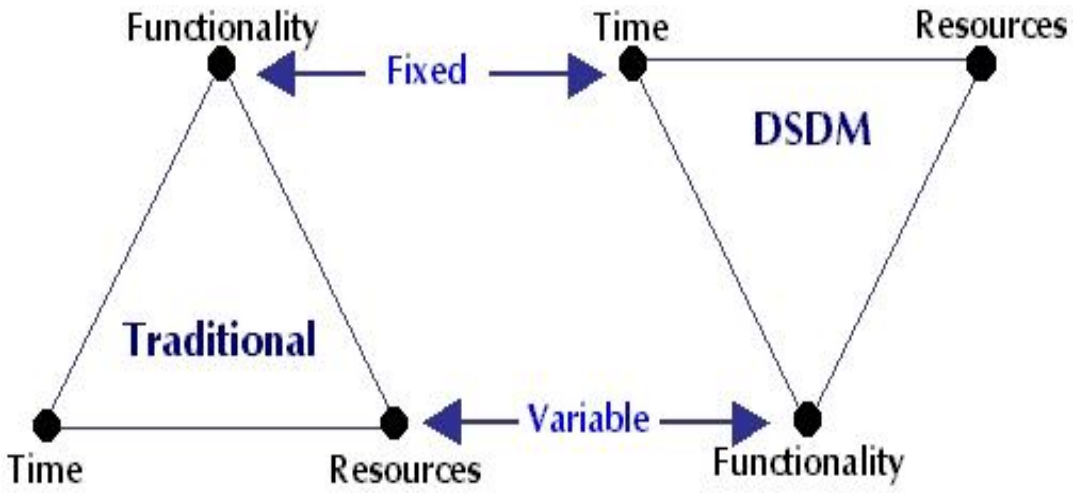
DSDM Konsorsiyum tarafından önerilmiştir (www.dsdm.org).

DSDM—ayırt edici özellikleri

- Pek çok yönden XP ve/veya ASD’ye benzer
- Dokuz yönlendirici prensip
  - Kullanıcıların aktif katılımı önemlidir.
  - DSDM takımları karar vermede yetkilidirler
  - Sık teslim edilen ürünlere odaklanılır.
  - Teslim edilebilirliğin kabul kriteri olarak iş amacına uygunluk kullanılır
  - Tekrarlanan (iterative) ve artımlı (incremental) geliştirme doğru iş sonucuna ulaşmak için gereklidir.
  - Geliştirme sırasında gerçekleştirilen tüm değişikler geri alınabilirdir..
  - Gereksinimler yüksek seviyede temellendirilirler
  - Test tüm yaşam döngüsü boyunca entegredir

Dinamik Sistem Geliştirme

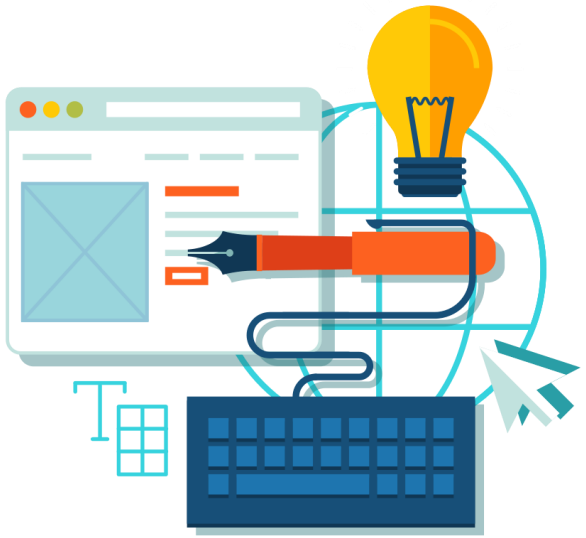
- Basit.
- Genişletilebilir.
- Düz ileri Framework tabanlı.
- Tüm proje türlerinde çözüm sağlayıcı değildir.



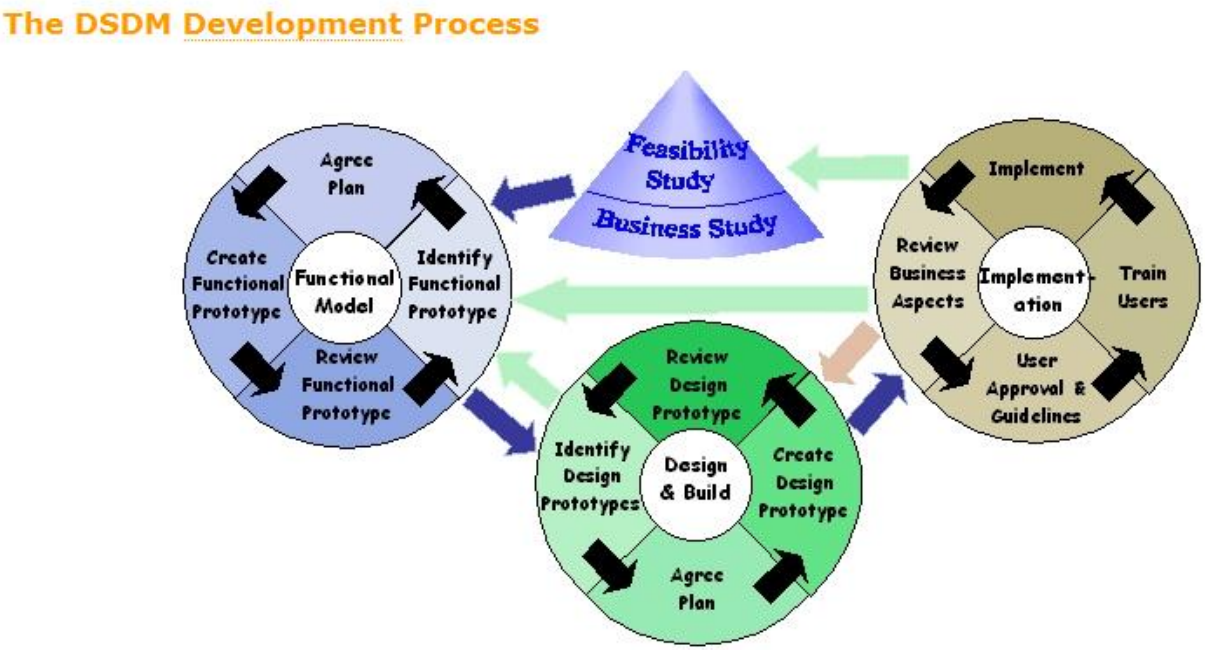
Dinamik Sistem Geliştirme Proje Yapısı

DSDM projesi, organize rolleri, gömülü rolleri ve sorumlulukları ile zengin bir küme ve çeşitli çekirdek teknikleriyle desteklenen 7 aşamalı adımdan oluşur.

- Roller ve Sorumluluklar
- Takım Organizasyonu ve Boyutu
- 7 Aşamalı kurallar. Bunlar:
  1. Ön Proje
  2. Fizibilite Çalışması
  3. İş Çalışması
  4. Fonksiyonel Model Yineleme
  5. Tasarım & Yapı Yineleme
  6. Uygulama
  7. Proje Sonrası



Dinamik Sistem Geliştirme Süreci



Scrum

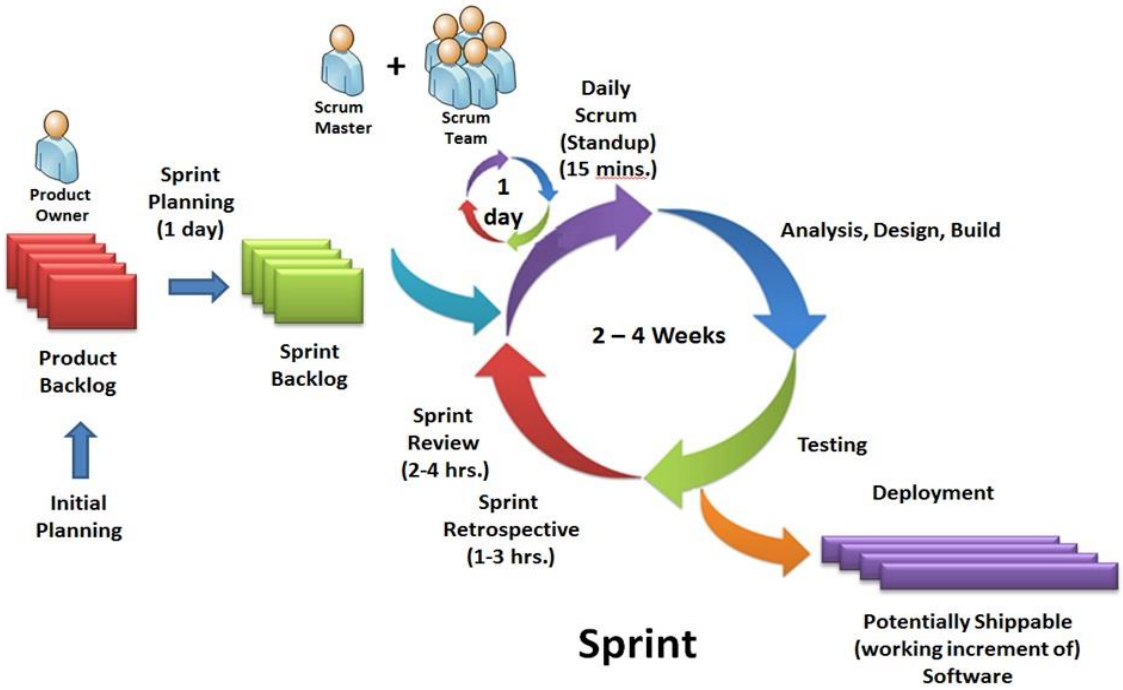
- Jeff Sutjerland ve Ken Schawaber tarafından 1990'ların ortalarında geliştirilen, proje yönetimi ve planlama ile ilgili yöntemlere odaklı olan ve mühendislik detayları içermeyen bir modeldir.
- Yazılımı küçük birimlere (sprint) bölerek, yinelemeli olarak geliştirmeyi öngörür.
- Bir yinelemenin tanımlanması 30 günden fazla sürmemekte ve günlük 15 dakikalık toplantılarla sürekli is takibi yapılır.
- Karmaşık ortamlarda adım adım yazılım geliştiren küçük ekipler (<20) için uygundur.
- Yüksek seviyede belirsizlik arz eden projelerde, son yıllarda daha yaygın biçimde uygulandığı ve başarılı sonuçlar ürettiği bildirilmiştir.

Scrum Ayırt Edici Özellikleri

Scrum

- Geliştirme işi “paket”lere bölümlenmiştir.
- Test ve belgelendirme ürün oluşturuldukça süregelendir.
- İş “sprint” ler halinde gerçekleştirilir ve var olan gereksinimlerin “backlog” u olarak çıkarılır.
- Toplantılar çok kısadır ve bazen başkan bile içermez.
- Zaman aralıklı olarak müşteriye “demo” lar sunulur.

Scrum Süreci



Kaynak: http://www.insolaria.it/blog/2014/04/scrum-e-le-medotologie-agili/

Scrum Rolleri

Şeffaflık

Sürecin tümleriyle geliştiriciye görünür olmasıdır. Böylece ortak bir görüş ortaya konur. Örneğin, "bitti" takımdakilerin artık işin sonuna gelindiğinde hep bir ağızdan söyledikleri bir ortak bir cümledir. Çalışılan sürecin bittiği anlamına gelir.

Denetim

Scrum, kullanıcılarına yönelik istenmeyen sapmaları tespit etmek için sürecin ve programın sürekli denetim altında tutulması gerekir. Scrum karmaşık ürün gelişimini basite indirgemek için yapılandırılmış bir çerçeve olarak düşünebiliriz

Uyum

Yazılan programların talebe uygun olup olmaması durumudur. Herhangi bir saptama söz konusu ise bu açıklar giderilir ve talebe uyumlu hale getirilir. Saptamaları aza indirmek kısa bir sürede yapılmalıdır.

Scrum Takımı

Ürün Sahibi, Geliştirme Ekibi ve Scrum Master'dan oluşur. Takım kendi kendini örgütler. Böylece kendi içerisinde uyum içinde olan takımlar daha başarılı sonuçlar alırlar. Scrum'ın takım modeli esneklik, yaratıcılık ve verimliliği optimize etmek için tasarlanmıştır.

Ürün Sahibi (Product Owner)

Ürünün değerinden ve gerikaydından (Backlog) sorumlu olan kişidir . Ürün Backlog öğeleri açık olmalıdır. Bunuda sağlayacak olan kişi Product Owner'dir. Sorumluluğun büyük bir kısmı Product Owner'dadır. Takımda Product Owner tektir. Product Owner'in başarılı olması için, tüm organizasyonun kendi kararlarına saygı göstermesi gerekir. Geliştirme Ekibi, Product Owner'in kararlarına göre hareket eder .

Scrum Rolleri

Sprint

Scrum çerçevesinde her bir iterasyon Sprint olarak isimlendirilmektedir. Sprintleri paketlenmiş zaman aralıkları olarak düşünebilirsiniz. Yeni Sprint önceki Sprint sonucundan hemen sonra başlar. Sprintlerimiz başlangıç ve bitiş tarihleri bulunmaktadır, ve ekibin hızını ayarlamak için her bir sprint'in aynı sürede tutulması önerilmektedir. Scrum yöntemi ile yönetilen projeler "sprint" denilen fazlara ayrılır. Her bir "sprint"in süresi genellikle (en fazla) 30 gündür. İdeal olan proje boyunca tüm "sprint"ler aynı süreli olur.

Sprint İnceleme

Burada katılımcılar, gelecek hakkında iş birliği yapabilir. Gayriresmi bir toplantı olup iş birliğini daha iyi gerçekleştirmek için toplanılır. Bu, bir aylık sprint için dört saatlik zamanı kapsar. Sprint İnceleme sonucu bir sonraki Sprint için muhtemel Ürün Backlog öğelerini tanımlar.

Sprint Planlama Toplantısı

Sprint ile yapılacak çalışma Sprint Planlama Toplantısı ile planlanmıştır. Bu plan tüm Scrum Ekibinin ortak çalışmasıyla oluşturulur. Örneğin, iki haftalık Sprint'te dört saatlik Sprint Planlama Toplantıları vardır. Bu toplantılarda artımın nasıl sağlanacağı konuşulur.

Günlük Scrum (Daily Scrum)

15 dakikalık toplantılardan oluşur. Bu günlük Scrum'larda planlama yapıldığı gibi bir sonraki gün için tahminler yapılır. Bu toplantılar her gün düzenlenir ve amaç karmaşıklığı azaltmaktır. Bu toplantılarda her bir üye aşağıdaki sorulara cevap verir :

- Son toplantıdan bu yana neler tamamlandı ?
- Sonraki toplantıda neler yapılacak ?
- Engeller ne şekildedir ?

Kısaca hedefe doğru ilerlemeler değerlendirilir.

Scrum Rolleri

Sprint Backlog

Takım belirlenir ve “Sprint Planning” dediğimiz en fazla 4 hafta sürecek olan küçük döngülerle “Sprint” ile işe başlanır. Her döngü için ürün gerikaydından önemli gereksinimler seçilerek sprint gerikaydı (Sprint Backlog) oluşturulur ve sprint boyunca bu gereksinimler geliştirilir.

Product Backlog

İlk önce müşteriye yani ürün sahibine (Product Owner), istediği ürün gereksinimlerinin neler olduğu sorulur ve bu ihtiyaçlar çıkartılır (Product BackLog). Product BackLog içerisinde maddeler en önemlisinden en aza doğru sıralanır. Belirli bir periyot içerisinde belirtilen gereksinimleri karşılayan tam olarak bir portatif müşteriye teslim edilir .

Sprint Retrospective

Sprintte bir sonraki aşama için iyileştirme fırsatı yakalanır. Bu bir aylık sprint için toplamda 3 saatlik toplantıdır. Burada takım işini kolaylaştırmak için planlar oluşturulur. Böylece gelişim süreci ve sonraki Sprinti daha etkili ve zevkli hale getirmek için uygulamalar geliştirme ortamı yaratılır. Bu toplantılar sonucunda ürün kalitesini arttırma yolları planlanmış olur.

Scrum Rolleri

Scrum Ustası(The Scrum Master)

Scrum'dan sorumlu olan kişidir. Scrum Master, Scrum Takım teorisi ve uygulamaları kurallara uyarak sağlar. Scrum Master takım için liderdir. Scrum Master, ekip tarafından yaratılan değeri maksimize etmek için çalışır. Scrum Master, çeşitli yollarla Ürün Sahibine yardım eder . Bunlar:

- Ürün Backlog yönetimi bulma teknikleri
- Geliştirme ekibinin vizyon ve hedeflerini belirlemek
- Açık ve özlü Ürün Backlog öğeleri oluşturmak için geliştirme ekibine temel yolların öğretilmesi
- Anlaşılabilir bir ortamda uzun vadeli ürün planlamaları yapmak
- Çevikliği anlama ve uygulama

Scrum Master, geliştirme ekibinde hizmet eder:

- Örgütlenmeye koçluk eder
- Geliştirme ekibi için değeri yüksek ürünler oluşturmak
- Geliştirme ekibinin ilerlemesine engel olacak şeyleri ortadan kaldırmak
- Talebe göre Scrum olaylarının kolaylaştırılması

Scrum Master organizasyon içinde çalışır:

- Scrum'ın belirlenmesinde koçluk eder
- Örgüt içinde planlama yapar
- Takımın verimliliğini arttırır

Scrum Örnek

Firma bir proje alıyor ve proje tarihi sabit değil. Proje sahibinin beklentileri yüksek ancak istekleri yeterince net değil. Ayrıca süreçler içinde yeterli zaman bulunmamakta. Proje ekipleri arasında iletişim problemleri de var. Proje ise Kredi Kartı Taahhüt Programı. Sonunda iş birliği ile 8 ay olarak yer alan proje, 3'er haftalık 6 Sprint ile 18 haftada tamamlanıyor.

Özellik Güdümlü Gelistirme (Feature-Driven Development – FDD)

- Jeff De Luca ve Peter Coad tarafından 1997'de geliştirilen ve özellik tabanlı gerçekleştirimi esas alan bir modeldir.

- Süreç beş ana basamaktan oluşur; her bir basamak için çıkış şartları tanımlanır ve bunlara uyulur.
  - Genel sistem modelinin geliştirilmesi,
  - Özellik listesinin oluşturulması,
  - Özellik güdümlü bir planlama yapılması,
  - Özellik güdümlü tasarımın oluşturulması,
  - Özellik güdümlü geliştirmenin yapılması.

- Sisteme yeni bir özellik kazandırılmadan önce, detaylı bir tasarım çalışması yapılarak bu özelliği kapsayan mimari yapı oluşturulur.

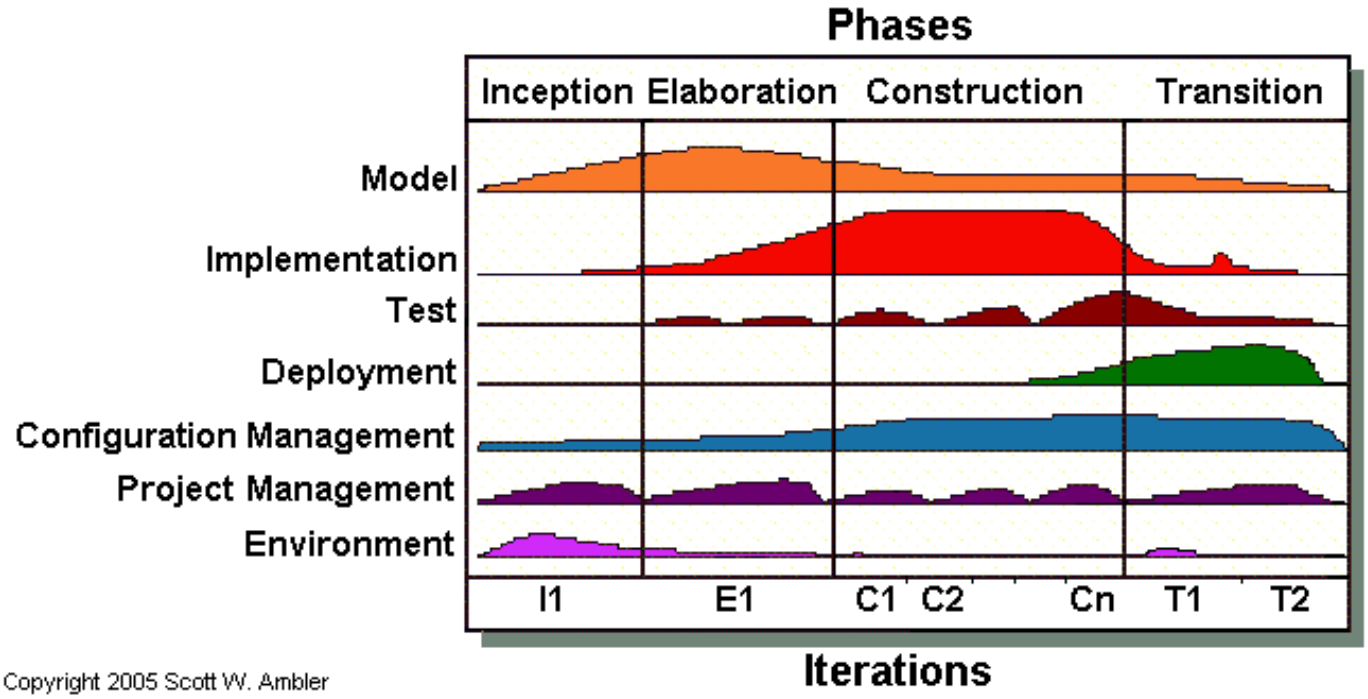
Çevik Tümlesik Süreç (Agile Unified Process – AUP)

Çevik Tümleşik Süreç (Agile Unified Process – AUP"); Rational Unified Process (RUP)'un, çevik yaklaşıma göre adapte edilerek basitleştirilmiş halidir.

Test-güdümlü geliştirme ("test driven development – TDD"), çevik değişiklik yönetimi, veri tabanını yeniden yapılandırma gibi çevik pratikleri uygulatır.

RUP'dan farklı olarak, 7 adet disiplin içerir.

Çevik Tümlesik Süreç



Yazılım Süreç Modeli Seçimi

- Yazılım süreç modelleri birbirinden tümüyle ayrı değildir ve çoğu zaman aslında beraber kullanılırlar.

- Hangi modelin ve model içerisindeki hangi adımların gerçekleştirileceğini seçmekle görevli olan kişiye “süreç mimarı” denir.

- Her modelin güçlü ve zayıf yanlarını değerlendirdikten sonra süreç mimarı proje için en iyi modeli seçmelidir.

Yazılım Süreç Modeli Seçimi

Süreç modeli seçiminde yararlı olabilecek bazı kriterler;

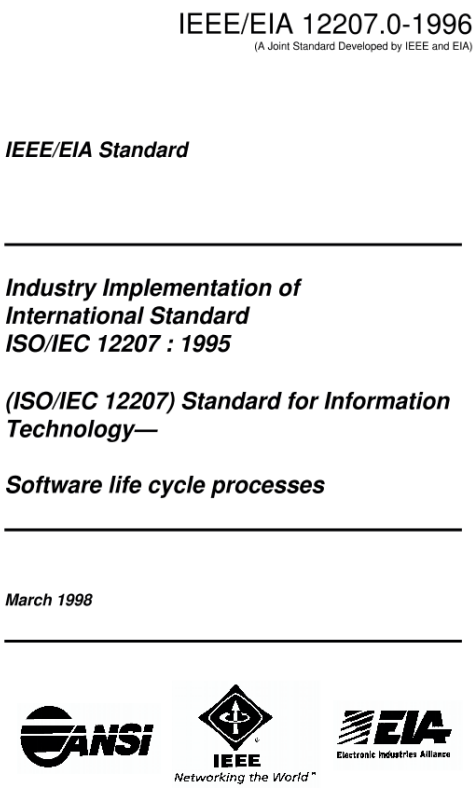
- Modelin oluşabilecek riskleri tolere edebilme kapasitesi
- Geliştirici kurumun son kullanıcılara erişim imkanı
- Bilinen gereksinimlerin ne kadar iyi tanımlanabildiği
- Erken işlevlerin önemi
- Problemin karmaşıklığı ve çözüm için olası adaylar
- Gereksinim değişikliğinin tahmin edilen sıklığı ve oranı
- Kurumun yönetimsel kapasitesi

Yazılım Süreçleri – IEEE/IEA 12207 nedir?

- Bilgi Teknolojileri-Yazılım Yaşam Döngüsü Süreçleri için Standard.
- Bir yazılım sisteminin tüm yaşam döngüsünü kapsayan süreçler kümesini tanımlar.
  - Kavramsal fikrin ortaya çıkışından
  - Yazılımın emekli oluşuna kadar
- Yazılım süreç modelleri için ortak bir çerçeve sağlar
- Bir organizasyon kendi iç kullanımı için ya da birden çok organizasyon kontrata bağlı çalışmak için kendine adapte edebilir.
- Projeye göre uydurulabilir
  - Genel, büyük ve karmaşık projeler için yazılmıştır
  - İhtiyaç, büyüklük, karmaşıklık, maliyet, zaman ve performansa uydurulabilecek şekilde tasarlanmıştır.
- Uygulamalar için bir kılavuzdur

12207 ne değildir?

- Yazılım geliştirme için bir reçete değildir.
- Yönetim ya da mühendisliğin yerine geçmez.
- Ürün ya da ölçme standardı da değildir.



Kullanımı

- Birincil yaşam döngüsü süreçleri:
  - Acquire, supply, develop, operate, and maintain software – Yazılım kazanma, tedarik, geliştirme, işletim ve sürdürme.
- Destekleyici yaşam döngüsü süreçleri:
  - Yukarıdaki fonksiyonları, kalite güvencesi, konfigürasyon yönetimi, ortak gözden geçirme, denetleme, geçerleme, doğrulama, problem çözme ve belgelendirme ile destekler.
- Kurumsal yaşam döngüsü süreçleri:
  - Organizasyonun süreçleri ve personelini yönetmesini ve geliştirmesini sağlar.
- Yazılım ürün yaşam döngüsünde yer alan müşteri, sağlayıcı ve tüm iştirakçiler arasında anlaşmayı artırır.

Yaşam döngüsünü bölümlendirme

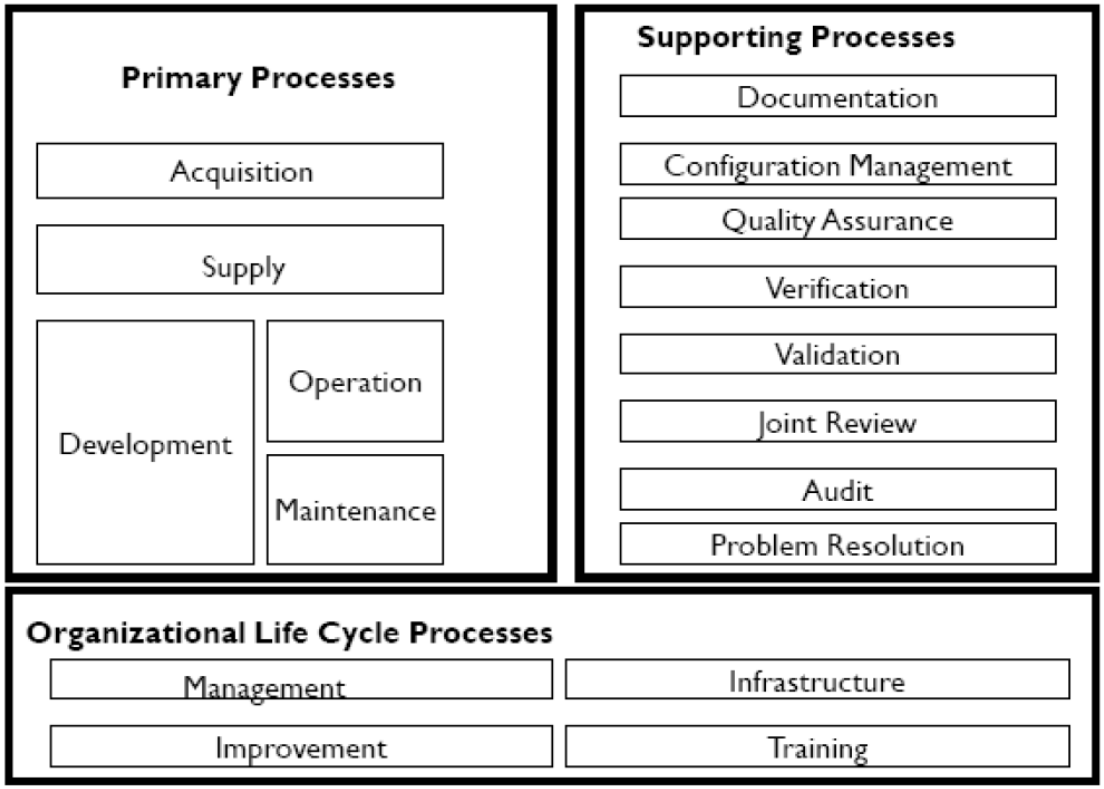
Modularity (modülerlik)

- Strongly cohesive (güçlü bağlı): süreç içindeki işler (task) fonksiyonel olarak birbirine bağlı olmalı
- Loosely coupled (zayıf birleşme) Süreçler arasındaki bağlar en az olmalı
- Association rules (İlişkilendirme kuralları)
  - Bir fonksiyon birden fazla süreç tarafından kulanılıyorsa, o durumda fonksiyon kendi başına bir süreç haline gelir
  - Eğer süreç A sadece ve sadece süreç B tarafından çağrılıyorsa, süreç A B'ye aittir.

Sorumluluk

- Bir süreç yazılım yaşam döngüsündeki bir organizasyon ya da bir tarafın sorumluluğuna verilir
  - Parçaları farklı organizasyonların sorumluluğunda olan fonksiyonlar süreç olamaz.

IEEE/EIA 12207 Yaşam Döngüsü (Yazılım Süreçleri)



Uygulama

- Bir BS(Bilgisayar Sistemi) organizasyonunda proje yöneticisi olarak görevlendirildiniz.

- İşiniz daha önce geliştirdiklerinize benzer bir sistem geliştirmek olan sözleşmeli bir projeyi yönetmek. Yalnız bu defaki proje daha öncekilere göre daha büyük ve daha karmaşık.

- Gereksinimler müşteri tarafından baştan sona belgelendirilmiş.

➡ **Hangi süreç modelini kullanırsınız? Neden?**

Özet

- Birleşik Süreç 3 aşamalıdır. Bunlar; yinelemeli, arttırmalı ve evrimsel, risk güdümlü şeklindedir.
- Birleşik Süreç Fazları başlangıcından üretime kadar 7 aşamada iterasyon yapılarak gerçekleştirilir.
- Çevik modeller, mevcut geleneksel modellere alternatif olarak, 1990'larda ortaya çıkmaya başlamıştır.
- Çevik (Agile) Yaklaşım da daha önemliden az önemliye doğru bir gerçekleştirim mevcuttur.
- Çevik (Agile) Yazılım Süreç Modelleri 7 ana başlık altında toplanmıştır.
- 2001 yılında, dünyanın önde gelen çevik modellerinin temsilcilerinin çıkardıkları Çevik Yazılım Geliştirme Manifestosuna göre;
  - Bireyler ve aralarındaki etkileşim, kullanılan araç ve süreçlerden;
  - Çalışan yazılım, detaylı belgelerden;
  - Müşteri ile işbirliği, sözleşmedeki kesin kurallardan;
  - Değişikliklere uyum sağlayabilmek, mevcut planı takip etmekten; daha önemli ve önceliklidir.

Özet

- Çevik Yazılım Geliştirme Manifestosu- Prensipleri projenin baştan sona müşteri ile çalışanlar arasında gerçekleştirilip, minimum sürede tamamlanmadı öngörülmektedir. Eğer müşteri istekleri değişirse yine de gereken zaman diliminde yapılanma hızlı bir şekilde gerçekleşmektedir.
- Uçdeğer Programlama (Extreme Programming – XP) 4 prensip etrafında toplanır. (Basitlik, İletişim, Geri bildirim, Cesaret)
- Uçdeğer Programlama 12 temel pratiğin birlikte uygulanmasını gerektirir.
- Adaptif Yazılım Geliştirme kurgu, işbirliği ve öğrenmeyi temel almaktadır.
- Dinamik Sistem Geliştirme Dokuz yönlendirici prensibi benimsenmiştir.
- DSDM projesi, organize rolleri, gömülü rolleri ve sorumlulukları ile zengin bir küme ve çeşitli çekirdek teknikleriyle desteklenen 7 aşamalı adımdan oluşur.

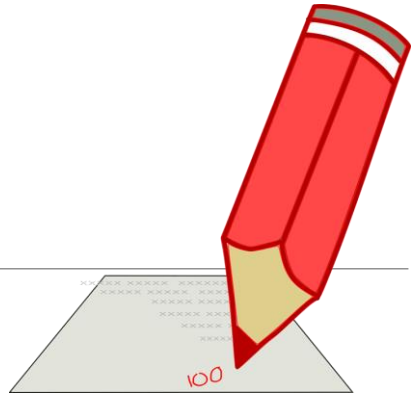
Özet

- Scrum, Jeff Sutjerland ve Ken Schawaber tarafından 1990'ların ortalarında geliştirilen, proje yönetimi ve planlama ile ilgili yöntemlere odaklı olan ve mühendislik detayları içermeyen bir modeldir.
- Scrum, yazılımı küçük birimlere (sprint) bölerek, yinelemeli olarak geliştirmeyi öngörür.
- Scrum, Karmaşık ortamlarda adım adım yazılım geliştiren küçük ekipler (<20) için uygundur. İş, "sprint" ler halinde gerçekleştirilir ve var olan gereksinimlerin "backlog" u olarak çıkarılır. Zaman aralıklı olarak müşteriye "demo" lar sunulur.
- Scrum süreci önemli bir model olmakla birlikte rolleri bir projede olması gereken en belirleyici kısımlardır.
- FDD Süreci beş ana basamaktan oluşur; her bir basamak için çıkış şartları tanımlanır ve bunlara uyulur. Sisteme yeni bir özellik kazandırılmadan önce, detaylı bir tasarım çalışması yapılarak bu özelliği kapsayan mimari yapı oluşturulur.
- Çevik Tümleşik Süreç (Agile Unified Process – AUP"); Rational Unified Process (RUP)'un, çevik yaklaşıma göre adapte edilerek basitleştirilmiş halidir. RUP'dan farklı olarak, 7 adet disiplin içerir.
- IEEE/IEA 12207 Bilgi Teknolojileri-Yazılım Yaşam Döngüsü Süreçleri için bir Standarttır. Yazılım süreç modelleri için ortak bir çerçeve sağlar

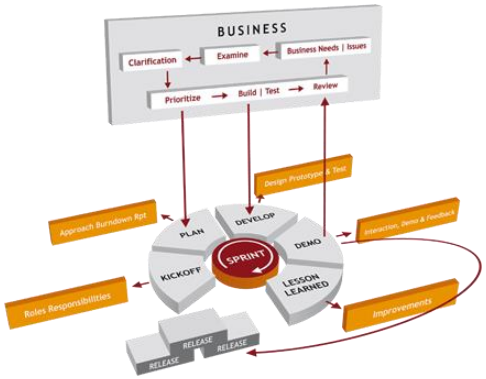
Çalışma Soruları

1. Birleşik süreç kavramı ve aşamaları nelerdir?
2. Birleşik süreç fazları nelerdir söyleyiniz?
3. Çevik modeller alternatif modellere istinaden kaç yılında ortaya çıkmıştır?
4. Çevik yazılım da nasıl bir gerçekleştirim vardır?
5. Çevik yazılım süreç modelleri kaç aşamalıdır açıklayınız?
6. Çevik modellerin temsilcilerinin çıkardıkları Çevik Yazılım Geliştirme Manifestosunu neyi amaçlamaktadır?
7. Uçdeğer Programlamanın benimsediği 4 temel prensibi söyleyiniz?
8. Adaptive Yazılım Geliştirmenin temel aldığı modelleme kavramları nelerdir?
9. DSDM projesi kaç aşamadan oluşur bunlar nelerdir?
10. Scrum nedir? Scrum modelini açıklayınız.
11. Scrum rolleri ne amaçla faaliyet göstermektedir. Bu rolleri söyleyiniz .
12. FDD Süreci aşamaları nelerdir söyleyiniz?
13. IEEE/IEA 12207 ne anlama gelmektedir?

Ödev



1. CRYSTAL modelini araştırınız.
2. Feature Driven Development modelini araştırınız.
3. Çevik bilgi Metodu (Agile Data Method) araştırınız.
4. Rational Unified Process (RUP)'i araştırınız.



Kaynaklar

“Software Engineering A Practitioner’s Approach” (7th. Ed.), Roger S. Pressman, 2013.

“Software Engineering” (8th. Ed.), Ian Sommerville, 2007.

“Guide to the Software Engineering Body of Knowledge”, 2004.

” Yazılım Mühendisliğine Giriş”, TBİL-211, Dr. Ali Arifoğlu.

”Yazılım Mühendisliği” (2. Basım), Dr. M. Erhan Sarıdoğan, 2008, İstanbul: Papatya Yayıncılık.

Kalıpsiz, O., Buharalı, A., Biricik, G. (2005). Bilgisayar Bilimlerinde Sistem Analizi ve Tasarımı Nesneye Yönelik Modelleme. İstanbul: Papatya Yayıncılık.

Buzluca, F. (2010) Yazılım Modelleme ve Tasarımı ders notları (http://www.buzluca.info/dersler.html)

Hacettepe Üniversitesi BBS-651, A. Tarhan, 2010.

Yazılım Proje Yönetimi, Yrd. Doç. Dr. Hacer KARACAN

http://www.cclub.metu.edu.tr/bergi_yeni/e-bergi/2008/Ekim/Cevik-Modelleme-ve-Cevik-Yazilim-Gelistirme

http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Fall_2011/ch6_6d_sk

http://dsdmofagilemethodology.wikidot.com/

http://caglarkaya.piquestion.com/2014/07/01/244/

Sorularınız