3. Hafta

YMÜ225
Yazılım Gereksinim ve Analizi

Dr. Feyza Altunbey Özbay

İçerik

- Çevik Yazılım Geliştirme
- Çevik Yöntemler
- Uç Programlama
  - İhtiyaç senaryoları
  - Yeniden üretim
  - Test
  - Eş Programlama

Çevik Yazılım Geliştirme

- Günümüzde şirketler küresel ve hızla değişen bir çevrede faaliyet göstermektedir. Şirketlerin yeni fırsatlara ve pazarlara, değişen ekonomik koşullara, ortaya çıkan rakip ürün ve servislere hızlı bir şekilde cevap vermeleri gerekir.
- Hızlı yazılım geliştirme ve teslimi birçok işletme için en kritik gereksinimlerden biri haline gelmiştir.

Çevik Yazılım Geliştirme

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

Bu nedenlerden dolayı hızlı yazılım geliştirme ve teslimine odaklanan geliştirme süreçleri gereklidir.

Çevik Yazılım Geliştirme

Hızlı yazılım geliştirme ihtiyacını ve değişen gereksinimleri ele alabilecek süreçler uzun yıllardır bilinmektedir.

Esas olarak 1990'ların sonunda

- Uç Programlama (Beck, 1999)
- Scrum (Schwaber and Beedle, 2001)
- DSDM: Dynamic Systems Development Method (Stapleton, 2003)

gibi «çevik yöntemler» fikirlerinin geliştirilmesi ile hız kazanmıştır.

Hızlı yazılım geliştirme, *çevik yazılım geliştirme* ya da *çevik yöntemler* olarak bilinir hale gelmiştir.

Çevik Yazılım Geliştirme

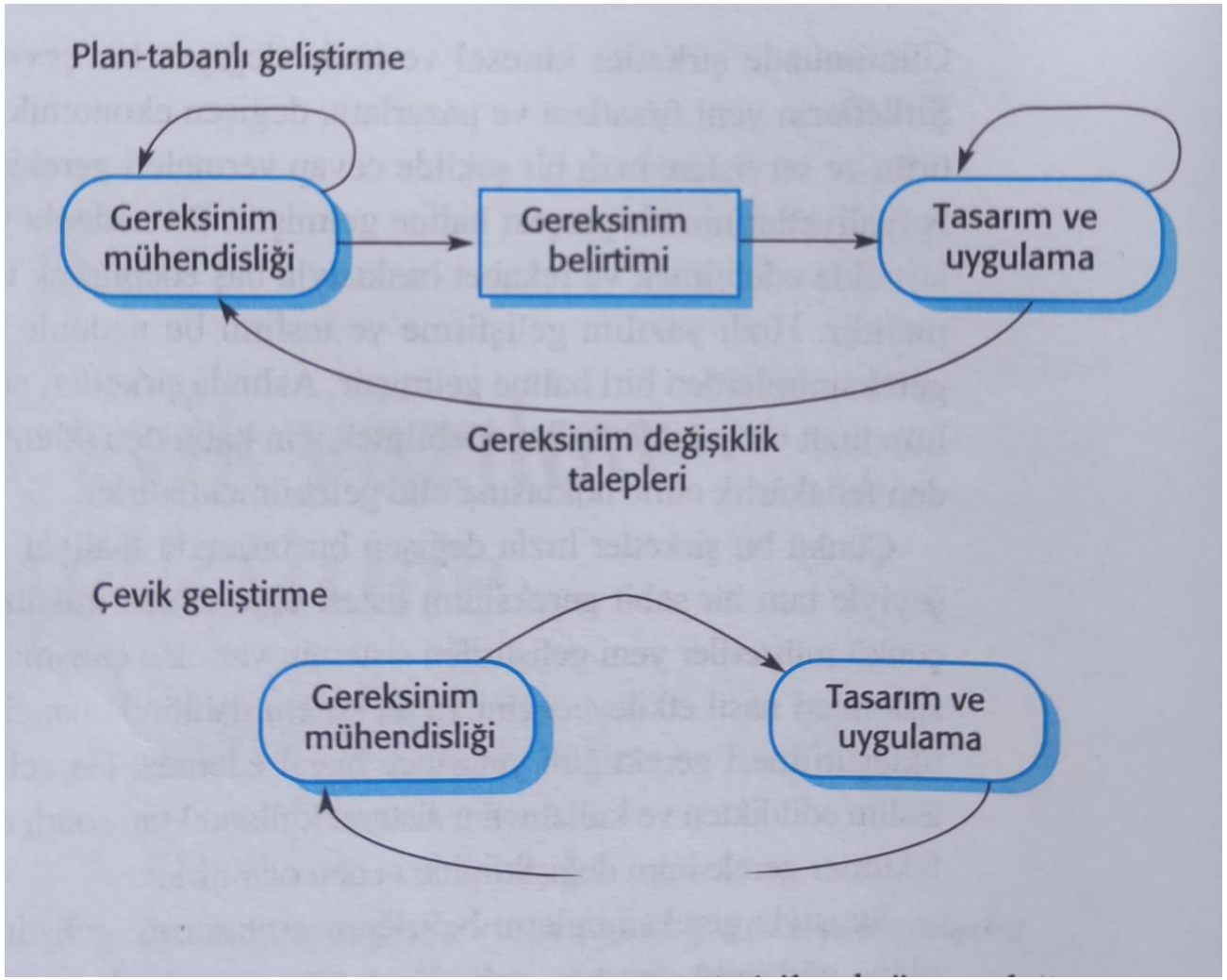
Çevik Yazılım Özellikleri

- Spesifikasyon, tasarım ve uygulama süreçleri iç içe geçmiştir.
- Sistem bir dizi artım ile geliştirilir.
- Son kullanıcılar ile sistemin diğer paydaşları her artımın belirlenmesi ve değerlendirilmesi süreçlerine katılırlar.
- Geliştirme sürecini desteklemek üzere kapsamlı araç desteği alınır.
- Değişen gereksinimler hakkında geri bildirim almak için müşteriler geliştirme sürecine dahil edilir.

Plan Tabanlı ve Çevik Geliştirme

- Plan tabanlı geliştirme
  - ✓Ayrı ayrı geliştirme aşamaları vardır. Her bir aşamada ne yapılacağı bellidir.
  - ✓Yalnızca çağlayan modeli değil. Artırımlı model de olabilir.
- Çevik geliştirme
  - ✓Tanımlama, tasarım, geliştirme ve test aşamaları iç içe geçmiştir ve geliştirme sürecinin çıktıları, yazılım geliştirme esnasında karşılıklı müzakere ile belirlenir.

Plan Tabanlı ve Çevik Geliştirme



Plan Tabanlı ve Çevik Geliştirme

- Çoğu proje plan tabanlı ve çevik süreçlerden unsurlar içerir. Bunlar aşağıdakilere bağlıdır:
- Gerçekleştirme aşamasından önce çok detaylı tanımlama ve tasarım gerekliyse plan tabanlı yaklaşım kullanılabilir.
- Yazılımı küçük sürümler halinde geliştirmek ve müşteriden geri dönüşleri almak mümkünse çevik tabanlı yaklaşım kullanılabilir.
- Geliştirdiğiniz sistemin büyüklüğü ne kadar? Çevik yöntemler, birbirleri ile yakın ilişki içerisinde olan ve bir arada bulunan geliştirme ekipleri için uygundur. Büyük çaplı projelerde bu mümkün olmayabilir.

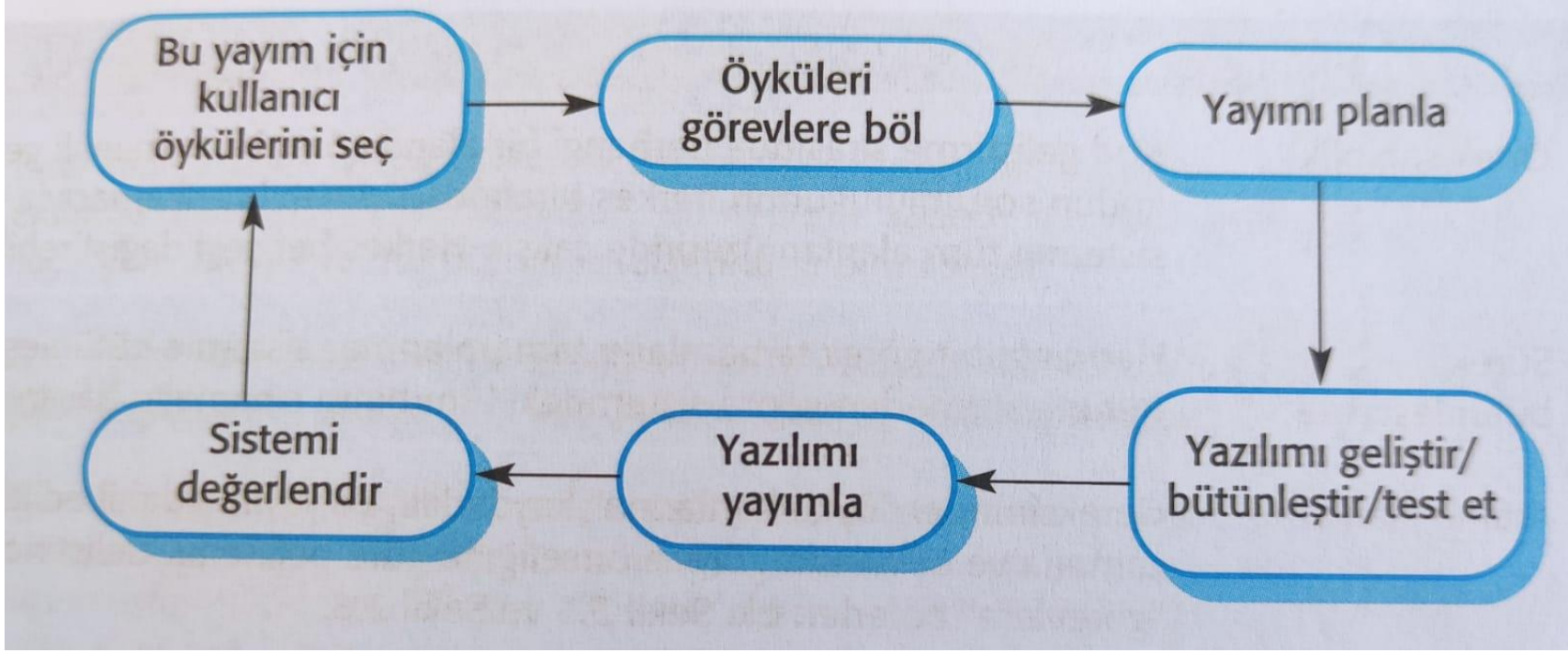
Çevik Yöntemler

Tüm çevik yöntemler yazılımın artırımlar halinde geliştirilmesi ve teslim edilmesi konusunda hemfikirdir. Bu nedenle birçok ortak ilkesi vardır:

İlke/prensip	Tanım
Müşterinin katılımı	Müşteriler geliştirme süreci boyunca geliştirme takımı ile yakın biçimde çalışmalıdır. Rolleri yeni gereksinimleri sağlamak, önceliklendirmek ve sistem artımlarını değerlendirmektir.
Değişimin benimsenmesi	Sistem gereksinimlerinin değişimi kaçınılmazdır, sistem bu değişiklikleri kabul edecek şekilde tasarlanmalıdır.
Artımlı teslim	Yazılım müşterinin her artım için belirlediği gereksinimlerin dahil edildiği artımlarla geliştirilir.
Sadeliğin sürdürülmesi	Hem geliştirilen yazılımda hem de yazılım geliştirme sürecinde sadeliğe odaklanın. Mümkün olduğu sürece sistemin karmaşıklığını azaltmaya çalışın.
Süreçler değil, insanlar	Yazılım geliştirme takımının yetenekleri bilinmeli ve bu yeteneklerden faydalanılmalıdır. Takım üyeleri detaylı tarif veren süreçleri takip etmek yerine kendi çalışma yaklaşımlarını geliştirme konusunda özgür bırakılmalıdır.

Uç Programlama

- Kent Beck (1998) tarafından geliştirilmiştir.



Uç program yayım döngüsü

Uç Programlama

- Küçük ve sık sistem sürümleri sayesinde artırımlı yöntem benimsenir.
- Müşterinin geliştirme sürecine katılımının anlamı müşterinin tam zamanlı olarak ekibe dahil olmasıdır.
- Süreç değil insan desteklenir. Bunun için pair programming(eş olarak program geliştirme) yöntemi kullanılır.
- Değişiklikler, rutin sistem sürümleri ile gerçekleştirilir.
- Kodun basitliği refactoring ile korunur.

Uç Programlama Pratikleri

İlke ya da pratik	Tanım
Ortak sahiplik	Kod geliştirme sırasında herhangi bir alanda özel bir uzmanlık gelişmemesi ve tüm kodun sorumluluğunun herkes tarafından alınabilmesi amacıyla yazılım geliştiriciler sistemin tüm alanları üzerinde çalışır. Herkes her şeyi değiştirebilir.
Sürekli bütünleştirme	Herhangi bir görev tamamlanır tamamlanmaz sistemle bütünleştirilir. Böyle bir bütünleştirmeden sonra sistemdeki tüm birim sınamalar başarılı olmalıdır.
Artımlı planlama	Gereksinimler "öykü kartlarına" kaydedilir, ve yayıma dahil edilecek öyküler, eldeki zamana ve öykülerin görece önceliğine göre belirlenir. Geliştiriciler bu hikayeleri "görevlere" bölerler. bk. Şekil 3.5 ve Şekil 3.6.
Her zaman hazır müşteri	Sistemin son kullanıcı temsilcisi (müşteri),UP takımına destek vermek amacıyla tam zamanlı olarak hazır bulunmalıdır. Bir U programlama sürecinde müşteri geliştirme takımının bir parçasıdır ve takıma geliştirme amacıyla sistem gereksinimleri iletmekten sorumludur.
Eş programlama	Yazılım geliştirenler birbirinin işini kontrol eden ve işi daha iyi yapabilmek için birbirine destek olan eşler halinde çalışır.
Kodun yeniden üretilmesi	Tüm yazılım geliştiriciler kodu sürekli olarak yeniden üreterek iyileştirmelidir. Bu, kodu sade tutar ve sürdürülebilirliği sağlar.
Sade tasarım	Mevcut gereksinimleri karşılayacak yeterli düzeyde tasarım geliştirilir daha fazlası değil.
Küçük yayımlar	Bir iş değeri sağlayan minimum fonksiyonellik öncelikli olarak geliştirilir. Sistem sık aralıklarla yayımlanır ve her artımda ilk yayımın üzerine yeni fonksiyonlar eklenir.
Sürdürülebilir hız	Fazla miktarda fazla mesai, kod kalitesinde düşüşlere neden olacağı ve orta vadede üretkenliği azaltacağı için kabul edilebilir değildir.
Test önce geliştirme	Fonksiyonun kendisi geliştirilmeden testini geliştirmek için otomatik bir birim test çerçevesi geliştirilir.

İhtiyaç senaryoları

- Kullanıcı ihtiyaçları senaryo ya da kullanıcı hikayesi olarak kartlara yazılır.
- Geliştirme ekibi bu kartları kullanarak ihtiyaçları görevlere böler. Bu görevler zamanlama ve maliyet tahminine temel teşkil eder.
- Müşteri, hikayeler arasından sonraki sürümde olmasını istediğini seçer

«Reçete Yazma» Örneği

Reçete yazma

Kate kliniğe gelen bir hastasına reçete yazmak isteyen bir doktordur. Hasta kaydı ekranda önceden görüntülenmiştir. Bu nedenle sadece ilaç alanına tıklar ve 'mevcut ilaç', 'yeni ilaç' ya da 'formül' seçeneklerinden birini seçer.

Eğer 'mevcut ilaç' seçeneğini seçerse, sistem dozu kontrol etmesini ister. Eğer Kate dozu değiştirmek isterse yeni ilaç dozunu girer ve reçeteyi onaylar.

Eğer 'yeni ilaç' seçeneğini seçerse sistem Kate'in hangi ilacı yazacağını bildiğini varsayar. İlaç adının ilk birkaç harfini yazar. Sistem bu harflerle başlayan olası tüm ilaçları listeler. Kate gerekli ilacı seçer ve sistem seçilen ilacın doğru olup olmadığını sorar. Kate son olarak ilaç dozunu girer ve reçeteyi onaylar.

Eğer 'formül' seçeneğini seçerse, sistem onaylanan formülleri görüntülemek için bir arama kutusu gösterir. Kate buradan gerekli ilacı arar, seçer. Sistem seçilen ilacın doğru ilaç olup olmadığını sorar. İlaç dozunu girer ve reçeteyi onaylar.

Sistem her zaman, girilen dozun onaylanan değerler arasında olup olmadığını kontrol eder. Eğer değilse Kate'den değiştirmesini ister.

Kate reçeteyi onayladıktan sonra, tekrar kontrol için gösterilir. Gösterim sonrasında 'Onay' ya da 'Değiştir' butonlarından birini tıklar. Eğer 'Onay'a tıklarsa, reçete denetleme veri tabanına kaydedilir. Eğer 'Değiştir'i tıklarsa Reçete Yazma süreci tekrar eder.

«Reçete Yazma» için Örnek Görev Kartları

Görev 1: Reçetelenen ilacın dozunu değiştir

Görev 2: Formül seçimi

Görev 3: Doz kontrolü

Doz kontrolü, doktorun tehlikeli olacak şekilde düşük ya da yüksek doz yazıp yazmadığını kontrol etmek için alınan bir güvenlik önlemidir.

Genel ilaç ismi için formül no'su kullanarak, formülü ara ve önerilen maksimum ve minimum doz bilgilerini ekrana getir.

Reçetelenen dozu, minumum ve maksimum değerlere göre karşılaştır. Eğer doz aralığın dışında ise "doz çok yüksek ya da çok düşük" şeklinde hata mesajı göster. Eğer doz aralığın içinde ise "Onayla" butonunu aktif hale getir.

Uç programlama ve Değişim

- Yazılım mühendisliğindeki geleneksel yaklaşım «değişim için tasarla» yaklaşımıdır. Değişim için gerekli olan eforu harca, böylece yazılım yaşam süresince daha az maliyetli olur.
- Ancak, Uç Programlama böyle düşünmez. Uç Programlama için öngörülemeyen değişimler için zaman harcamak boşunadır.
- Bunun yerine refactoring (yeniden üretim) ile ilgilenir. Böylece değişiklik gerekli olduğunda yapması daha kolay olur.

Yeniden Üretme

- Programlama ekibi mümkün iyileştirmeler için kodu inceler. O anda gerekli olmasa bile bu iyileştirmeyi yaparlar.
- Yeniden üretme, programın anlaşılırlığını arttırır ve dokümantasyon ihtiyacını azaltır.
- Değişiklikleri yapmak kolaydır çünkü kod iyi yapılandırılmış ve anlaşılabilirdir.
- Ancak bazı değişiklikler mimari boyutunda refactoring ister. Bunu yapmak biraz pahalı olabilir

Yeniden Üretme

- Birkaç yeniden üretim örneği;
  - Sınıf hiyerarşisini yeniden organize ederek tekrar eden kodları elemek.
  - Fonksiyonları ve özellikleri yeniden adlandırarak daha anlaşılır hale getirmek.
  - Satır içi kodları, bağımsız fonksiyonlar haline getirip kütüphane oluşturmak

Özet

- Çevik yöntemler hızlı geliştirmeye odaklı, sık sık sürümler oluşturan, süreç yüklerini azaltan ve yüksek kaliteli kod üreten yöntemlerdir. Müşteriyi doğrudan geliştirme sürecinin bir parçası yaparlar.
- Plan tabanlı ya da çevik bir yöntemin seçimine karar vermek geliştirilecek yazılımın tipine, geliştirme ekibinin yeteneklerine, kurum kültürüne bağlıdır.
- Uç Programlama, en iyi bilinen çevik yöntemdir ve sık sık sürüm geliştirme, kodların sürekli iyileştirilmesi ve müşterinin geliştirme takımına katılımı gibi birçok uygulamayı (pratik yaklaşımı) barındırır.

Test

- Test, Uç Programlama ’da işin merkezindedir.
- Uç Programlama ’da test özellikleri:
  - ✓Önce-test yaklaşımlı geliştirme
  - ✓Artırımlı test geliştirme.
  - ✓Test ve doğrulama esnasında müşterinin katılımı.
  - ✓Yeni bir sürüm üretmeden önce bütün bileşenleri otomatik test araçları ile test etmek.

Müşteri Katılımı

- Müşterinin testlere katılması kabul testinin geliştirilmesine yardımcı olur.
- Müşteri, testi yazan ekibin bir parçasıdır. Böylece, geliştirilen her

yeni kodun müşteri ihtiyaçlarını karşıladığından emin olunur.

«Doz Kontrolü» için Test Örneği

Test 4: Doz kontrolü

Girdi:

1. İlacın tek dozuna ait mg ile ifade edilen bir sayı
2. İlacın tek dozlar şeklinde günlük alınma sayısı

Testler:

1. İlacın tek doz miktarının doğru fakat günlük alım sıklığının fazla olduğu durumu test et.
2. İlacın tek dozunun yüksek ya da düşük olduğu durumları test et.
3. İlaç alım sıklığının yüksek ya da düşük olduğu durumları test et.
4. İlaç alım sıklığının izin verilen değerler içinde olduğu durumu test et.

Çıktı:

Onay mesajı ya da ilaç dozunun belirtilen değerler dışında olduğunu gösteren hata mesajı.

Uç Programlama – Test Zorlukları

- Programcılar test programı geliştirirken bazı şeyleri eksik bırakabilirler. Örneğin, olası bütün durumları kontrol etmeyebilirler.
- Bazı durumlarda testleri artırımlı olarak yazmak çok zordur. Örneğin, karmaşık bir kullanıcı arayüzüne sahip bir sistem için iş akışlarını gösteren birim testlerini yazmak gibi.
- Test edilen durumların eksiksiz olduğuna hüküm vermek biraz zordur. Birçok sistem testiniz olmasına rağmen bu testler olası bütün durumları kapsamıyor olabilir.

Eş Programlama

- Uç programlama kavramındaki yeniliklerden biri de programcıların yazılımı eşler halinde geliştirdikleri eş programlamadır.
- Eş programlamanın avantajları:
  - Sistem sorumluluğunun ortak olarak alınmasını sağlar.
  - Bilginin ekibe yayılımı artar.
  - Eş programlama esnasında bilginin yayılımı, ekip üyelerinden birinin ekipten ayrılması durumunda projenin başarısız olmaması için çok önemlidir.
  - Her bir satır kodun en az iki kişi tarafından kontrol edilmesini de garanti eder.
  - Yazılım yapısını güçlendirmek için yeniden üretimi teşvik eder.