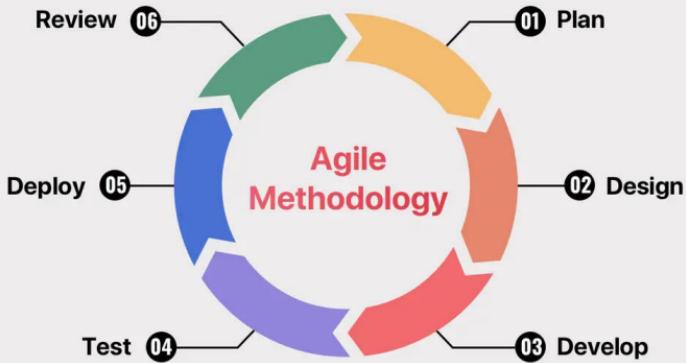


Barış Semeoglu

23.07.25

Istanbul

i2i Systems
Summer Internship



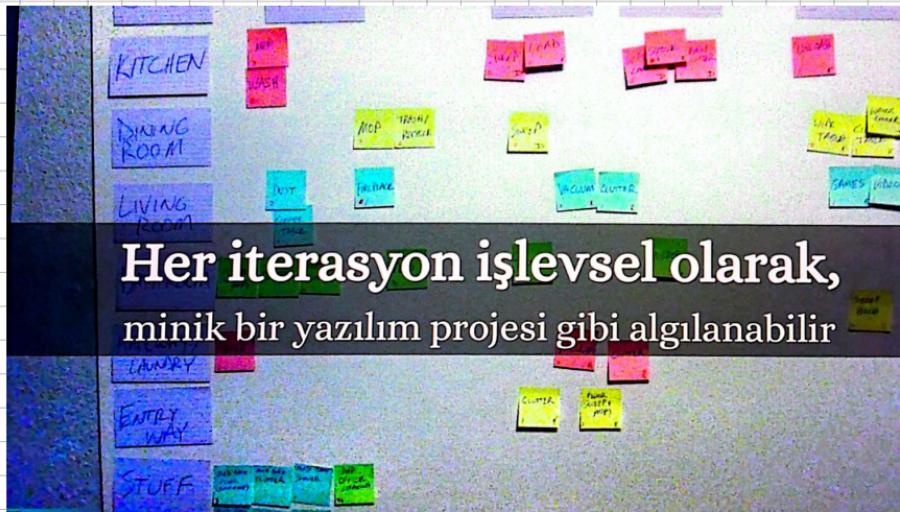
- Mühendislikle sistematikle, ölavzolabilir ve tarihi süreçlere dayalı olarak ün göstirmeyi hedefler
- Yazılım mühendisliği, yazılım ünvanı sistematik, şartsız ve ölavzolabilir bir şekilde davranışsal anlamına gelir

• İnstagram • Kalite • Kontrol sonmlar



- **Geçerleme yazılımda "time-boxing" kılavuzları Riske minhazize etmeye için 1 hafta ile 1 ay arası içinde**

Her time-boxing kılavuzu denetli aslaan bir ünvan parçası şebekeyi oluşturur



Gerrileme, değişime hızlı tepki verme yeteneği olarak tanımlanır.

Gerrileme yazılım: yazılım projelerinde kılavuzlar bir istekset işipi olarak forman.

- En Dergin olan Agile yöntemleri:

1- Extreme Programming (XP)

2- Scrum

3- Feature Driven Development

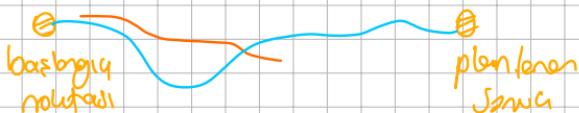
4- Lean Software Development

5 - Kanban

- agile yöntemler, yazılım tallarımlarının işflahlığı, yapısını değiştirmesi, projelerin doğası i̇ş gerekliliklerine yanıt verme能力和ası sağlar.

Agile ve Klasik Geliştirme

- the "Code and Fix" model



- yazılım öncesi müşterisi • talebin ile birlikte çalışır

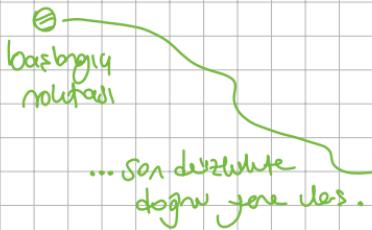
→ ince ölçütler
→ gün testimleştirmeler

- Heavyweight, Disciplined model



- yazılım öncesi müşteri
• talebin ile birlikte çalışır
• kodlama, test, değerlendirme
• bir model for olur
• - - - - -

- Adaptive, Lightweight (Agile) Process



- agile yöntemler ile yazılım geliştirme süreci müşterinin ihtiyaçlarını karşılar
• - - - - -
• - - - - -

• gerekçisini değirmenlerde uyabilecek bir langır

• en baştan her şeyi planlamakla

• değirmeni kıraklıyan bir zepi

agile yapısı

- Avantajları:

— Ünen kalitesine olan sorumluluğu vurgular

— Ortalı metafor: isi yalan ve net tonda

— Sistem içeriği iyileştirmelerde etkilendirilir

— Tüm testler iyileştirme sırasında iyileşen
yeniliklerin kapsamı hızla belirlenir

— müsteri, kapsam, öneelik ve tarihtene iş perspektifinden
korarır.

— Tadım iterasyonu: tahmin ve takip eðar

— incremental (artırmalı) geliştirmeye stratejisi uygulanır



"XP (Çevik Yöntemler), yazılım geliştirmek için hafif, verimli, düşük riskli, esnek, öngörelebilir, bilimsel ve eğlenceli bir yöntemdir." - Kent Beck, Extreme Programming Explained

XP Dezavantajları

• Döviz işbirliğine dayalı
yani gelen izlenme
kodu

• Tadım odaklı sistem
cole, kod odaklı sistem
cole

• Kantiþ testlerle sağlanır

• öncüllü tasarımın olasılığı

• öncüllü tasarımın kolay
uygulanması da

Agile manifesto

Daha iyi yazılım geliştirme yolculum uygulanır
ve bireylere de uygulamasına yardım etmek
için ortaya atılmıştır.

Bu çalışmaların sonucunda: Çevik Manifesto

- Süreçler ve araçlardan **ziyade** bireyler ve etkileşimlere
- Kapsamlı dokümantasyondan **ziyade** çalışan yazılıma
- Sözleşme pazarlıklarından **ziyade** müşteri ile işbirliğine
- Bir plana bağlı kalmaktan **ziyade** değişime karşılık vermeye daha fazla değer vermeye karar vermiştir.

Çevik manifesto, sol taraftaki maddelerin değerini kabul etmekle birlikte, sağ taraftaki maddeleri daha değerli bulmaktadır.

- *çokluşun şartum, iktidaronun bitiştiğinde ölçüsündür*
- *agile yöntemler i̇şerdilebilir gelişmeyi tercih etmektedir*
- *teknik mülakatlıyet ve şeritlin konumdaşı simdi i̇şten*
↳ *sadece işçilerdir* ↳ *İşçi işçileri dergileyen takımlar*

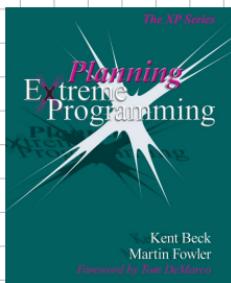
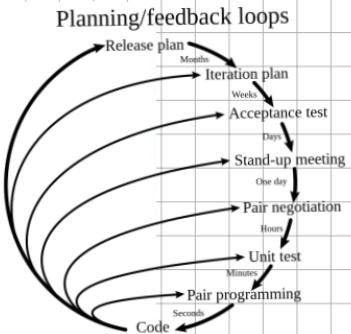
Extreme programming

- Yüksek kaliteli yazılım ürününün kısıtlı imkânlarla geliştirilebilmesi için düşünülmüştür.
- Hafif Siklet yazılım geliştirme süreci
 - **Dokümantasyonun** bir kısmı iletişim aktiviteleri ile değiştirmiştir
 - Kaynak koda ve testlere odaklıdır
 - **Yüksek üretkenlik iyileştirmesi**
- Endüstriyel **Pratisyenler** Tarafından Geliştirilmiştir
 - Bayerische Landesbank, Credit Swiss Life, DaimlerChrysler, First Union National Bank, Ford Motor Company and UBS."

• en önemli direktile
değerli şartumun erken
ve derinlik felsefemi
saygınlıkla

• değişim gerekimleri
her zaman dikkate
alınılmaktadır

- agile yöntemler, değerli
müzeticinin "rehabet"
çarşılığı için kullanılır
- matris çalışma bireyler
- just cause Meeting



XP – Zorluklar & Varsayımlar

- XP, yazılım mühendisliği ile diğer mühendislikler arasında kurulan analogilerin yanlış olduğunu iddia eder.
 - Yazılım müşterisi gereksinimleri **cok sık** değişirir
 - Yazılım ürünleri daha kolay **değiştirilebilir** olmalı.
 - Tasarım maliyeti: üretim maliyeti oranı değişkendir.
 - Kodlamayı tasarım derlemeyi de sürüm olarak kabul edersek.
 - Sürüm üretimi çok çabuk ve ucuz olmalıdır.
 - Yazılım Geliştirme bir tasarım aktivitesidir.

→ Tasarım *sadece belli olan gerekşim ile bulduğum, olsa getecele işlevselligi koronat.*

XP Temel Değerler

- Değerler çevik geliştirme kültürünün oluşması ve üretkenliğin artırılması için gereklidir.
 - İletişim
 - Geri besleme
 - Basitlik
 - Cesaret
- XP, temel çevik değerleri desteklemek ve güçlendirmek için, 3 gruba ayrılabilen bir dizi planlama, test ve geliştirme pratikleri önermektedir:
 1. Programlama pratikleri
 2. Takım pratikleri
 3. Proje pratikleri

XP'nin Temelleri

- İş paydaşları ve geliştiriciler tarafından verilen kararları birbirinden ayırmayı.
- Yalın düşünüp - tasarımını olabildiğince basit tutar. "Bugün için tasarıla yarın için değil"
- Üretim kodunu yazmadan önce test kodunu yazma ve tüm testleri sürekli çalışma.
- Eşli Programlama yapma
- Hızlı teslimat ile çok kısa yinelemeler

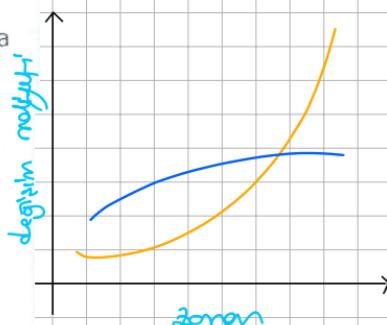
XP ile ilgili çekinceler

- Uzman yoktur.
 - Tüm yazılım geliştiriciler; mimari, tasarım, test ve entegrasyon süreçlerine katıllırlar.
- Öncül ayrıntılı analiz ve tasarım yapılmaz
- Altyapı için yapılacak ön çalışma kısıtlıdır
- Testler ve kodlamanın yanında tasarım ve uygulama raporlamasına ekstra zaman harcanmaz.

XP ne zaman uygundur?

- Küçük projeler ideal:
 - 5 veya 10 kişilik takımlar, 20 maksimum.
- Geliştirici ve müşteri temsilcisi aynı yerde olmalı.
- Problemler:
 - Bas konuş kültürü (point-click culture)
 - Testleri tekrar tekrar yapmak zaman maliyetli

- tüm doğruları, en iyi pratikleri uygleyerek maxima zaman
- takım içi emberi oluştur



- : geleneksel yöntem
- : extreme programming

Dört Önemli XP Değeri

- | | |
|--|---|
| <ul style="list-style-type: none">• Basitlik<ul style="list-style-type: none">-En basit çözüm en iyi çalışandır-İhtiyacın olmayacağına üretme.• İletişim<ul style="list-style-type: none">-Geliştiriciler-Kullanıcılar-Müşteriler-Testçiler-Kaynak kod | <ul style="list-style-type: none">• Geri besleme<ul style="list-style-type: none">-Test etme-Deneyelleştirme-Teslimat• Cesaret<ul style="list-style-type: none">-Güven-Birlikte geçmeşi |
|--|---|

Oniki XP Pratiği

1. Planlama Oyunu
2. Kısa aralıklı sürümler
3. Basit Tasarım
4. Test etme
5. Refactoring (kod düzenleme)
6. Eşli Programlama
7. Kolektif Mülkiyet
8. Sürekli Entegrasyon
9. Müşteri ile birlikte üretim
10. Sürdürülebilir Hız
11. Metafor
12. Kodlama Standartları

Planlama Oyunu

- Farkındalık
 - Proje başladığında her şeyi bilemeyez.
 - Müşteriler ne istediklerine şu anda kadar yaptığımız işlere bakarak karar verebilir (prototip, çalışan parça).
 - Geliştiriciler proje alan uzmanlıklarını ve teknoloji birimlerini proje yol alındıktan sonra iyileştirecekler.
 - Geliştiriciler ne soracaklarını proje başladıkten sonra öğrenecekler.
 - Başlangıçta neyi bilmeklerini de bilmiyor olabilirler.

- *cetvelizası garanti olan en günün gün geçmesini erte*
- *mükemmeliğin iş gerekliliklerini kesintiliyle korusla*
- *ilemeye yetenliklerin tasarımından mümkün olduğunda lütfen mümkün olduğunda lütfen*

1. Planlama Oyunu

• Sürüm Planlaması

- Üst düzey özellikleri 5-10 günlük iş paketleri olarak tanımlama ve efor tahmini.
- Müşteri ile birlikte sabit uzunluklu yineleme (iterasyon) Çizelgesine özellikler yerleştirilir

• Iteration (yineleme) Planlaması

- Aynı, ama sonraki iterasyon içinde 3 veya daha az gün içinde yapılabilecek detaylı hikaye kartları oluşturma.

Bu yöntemle projeyi başarıya taşımak kolaylaşmaktadır.

Planlama Oyunu

Kullanıcı Hikayesi

- Kartlara yazılır
 - müşterilerin bir şekilde oyundan etkilenmesi sağlanmalıdır
- Müşteriler tarafından iş degerine göre oncelik içeren ürün parçaları belirlenir
- Geliştiriciler tarafından tahmin 1, 2, 3 hattılık iş paketleri tamamlanır
- Geliştiriciler tarafından risk degeri tahmin edilir
 - Risk değerlendirmeleri yapmak teknik açıdan zor
- Kart, ayrıca yapılacak işler için bir tâahhutname görevi de görür

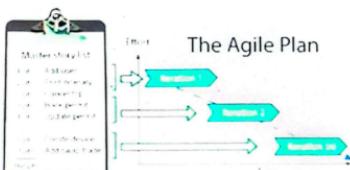
Örnek: Ben bir kütüphaneci olarak kitapları basım yılına göre arayabilmeliyim.

As a librarian, I want to be able to search for books by publication year.

2. Kısa aralıklı sürümler

İş parçalarını erken ve sıkça teslim et.

- Iterasyon tarihleri geçirilmelidir
 - Bir iterasyon sırasında zaman ve kaliteden asla ödün verilmez
- Küçük ve üretken takımlar kısa zaman dilimlerinde etkin tahminlerde bulunabilirler



4. Test yapma

- Her şey otomasyon ile alakalı
- Her kod parçası için otomatik birim testleri
- Her hikaye (gerekçinim) için otomatik kabul testleri oluşturulmalıdır.
- Tüm birim testleri, yeni bir özelliği ekledikten sonra %100 başarılı olmalıdır.



Refactoring (Kod Düzenleme) :

değişim zor olduğunda, değişikliklerin kolayca yapılmasına izin vermek için kod düzenleme yapılır, gerekli testleri koşturur ve ardından değişiklikler eklenir

- Değişim zor olduğunda, değişikliklerin kolayca yapılmasına izin vermek için kod düzenleme yapılır, gerekli testleri koşturur ve ardından değişiklikler eklenir



6. Eşli Programlama

- İki kişi çalışma, özellikle açık bir çalışma ortamında (ortak konum) tek olmaktan daha iyidir
- Erken kod denetimi
- Erken beyin fırçası
- Meslektaş baskıısı disiplini güçlendirir



Eşli Programlama

- Ekip arasında bilgi aktarımı, eşli programlama kullanıldığından büyük bir avantajdır
 - Her iki programcı da kod hakkında bilgi sahibidir
 - Kod yazmış ya da kod üretimini izleyen kişi aktif olarak katkı sağlamıştır.
- Geliştirme maliyetleri iki katına çıkmaz
- Sürekli eğitim imkanı sağlar

Testler

Birim Testleri

- Bize kodların çalışma durumunu gösterir.
- Geliştiriciler yeni kodu sürüm eklediklerinde birim testlerini çalıştırırlar
- Herhangi bir test başarısız olursa, sorunun nedenini irdelemek gerekir
- Sorun kesinlikle yeni eklenen bir şeyden kaynaklanmaktadır, çünkü testlerin % 100'ü son bir sürüm çıkmadan koşturulmuştur

Testler

Kabul Testleri

- Müşterinin gerekliliklerine dayalı test senaryoları
- Sistem uçtan uca test edilir
- Sistemin sahip olması gereken özelliklere sahip olduğunu müşteriye ve geliştiricilere gösterir
- "Proje Hızı" ölçmek için de kullanılan bir ilerleme yöntemidir

→ Müşterinin kabul testinin % kaççı çalıştırıldı?

fektörlerin parçalarının kırılımına hedeflenir

- *yazılım taraması* / geliştirime
- *kodun okunabilirliğini koruya* / geliştirme
- *hata bulunmayı* / geliştirme

- *kodda kütür yaşamaması*
- *gönderilen kod geçerliliği* / test
- *kodun okunabilirliği* / test
- *vân kod metotlarını* / geliştirme
- *vân parametreli döngüler* / türkçe

bu dumlarda refactoring yapılır

Eşli Programlama

- Geleneksel uzmanlaşma çevik yöntemde istenmez, herkes her işi yapmalıdır
- "Sürücü" uygular, kodlamaya odaklanır
- "Navigator" sorular sorarak sürücüyü yönlendirir ve yapıtları için açıklamalar bekler
- Sağlıklı eşleşmelerin her 45 ila 60 saniyede bir iletişime geçmemeleri beklenir
- Çiftler sıklıkla rol değiştirmelidir.
- Düşük karmaşılık gerektiren görevler için ayrı olarak da çalışabilirler
- Eşleme yoğun iş yükü getirir (her 2 saatte bir mola)

7. Kolektif Mülkiyet

- Değiştirilebilir programcılar (herkesin bir yedeği)
- Takım dört nala koşabilir
- Herhangi bir gecikme olmaksızın bir şeyler daha kolaylıkla değiştirilebilir



9. Ortamda Müşteri

- Müşteri / Kullanıcı yazılım ekibinin bir üyesidir
 - Detailli soruları yanıtlamak için öncelikler, açıklamalar için kullanılabilir
 - Gereksinimler hakkındaki programcı varsayımlarını azaltır
 - Paydaşlara neyi ödediklerini göstermek kolaylaşır

Yerinde müşteri ne sağlar?

- Geliştiriciler kararları beklemek zorunda kalmaz.
- Yüz yüze iletişim, yanlış anlaması olasılığını en aza indirir

12. Kodlama Standartları

- Tüm programcılar aynı formatta kodlamalı
- Nesnelerin birbirleriyle nasıl iletişim kurduğuna dair kurallar net olmalı.
- Neyi ve nasıl belgeleneceğine dair yönereler izlenmeli.



8. Sürekli Entegrasyon

- Entegrasyon problemleri daha küçük olacağından, her seferinde daha ufak problemler ile uğraşırız
- Böyleselike geleneksel ve yüksek riskli tam entegrasyon aşamasını ortadan kaldırılmış oluruz.



10. Sustainable Pace

- düzenli programlama daha çok hata yapar*
- verimli, olumlu ve etkili teknikler*
- en öne sürüldüğü*
- XP, UML ve dene yazılım geliştirme için sağılırlar*

11. Metaphor (mecaz)

- Bir "ad sistemi" kullanılır.
- Genel bir sistem açıklaması kullanılır
- Müşteriler, kullanıcılar, paydaşlar ve programcılar ile iletişim kurmaya yardımcı olur
- Bir metafor parçalarının toplamından daha büyük bir anlam yaratır ve bize yeni bir farkındalık sunar.
- Metaforlar, benzetmelerde olduğu gibi, anlatılmak istenen kavram, onunla bir yönden benzerliği olan başka bir kavramla anlatılmaya çalışır.

XP Uygulamaları Birbirini Destekler



XP'nin bazı avantajları

- Takım işbirliğini ve desteğini teşvik eder (eşli programlama)
- Değerlen / gelişen ihtiyaçlar için güçlü destek
- Mükemmellik için disiplin ve takım etğini geliştirir
- Kaliteyi geliştirir (kod yazılrken, kod denetimleri gerçekleştir)
- Gereksinimlerin başlangıçta tam olarak belirlenmeyeceği ve müşterilerin proje boyunca yoğun bir şekilde çalışmaya devam etmeleri gerektiği kabul eder (ortak sorumluluk).
- Sürrekli eğitim ve bilgi aktarımı sağlar
- Uzmanlaşmaları kontrol eder
- İlk önce test geliştirme (TFD) maliyet düşürür, projede erken inceleme sağlanmış olur.

XP'nin bazı kısıtları

- Müşteri gerçekçi bir şekilde geliştirme ortamında olabilir mi?
- Ekip üyelerinin aynı ortamda olmaları gereklidir, uzaktan çalışmaya pek uygun olmayabilir.
- Surecin olğulenebilirliği ile ilgili soru işaretleri:
 - Küçük takımlar → küçük projeler
- Sanal takımlar
 - XP ile yazılım geliştirme dağıtık ekipleler zor olabilir
- Tüm insanların eşit derecede etkili olabileceği kabulü
- Eşit bilgi düzeyinde takım elemanları bulmak ne kadar mümkün olabilir

11 Scrum Jöhteni ile Yazılım Geliştirme

Scrum – Genel Bakış

- En yüksek **iş değerini** en **kısa zamanda** sunmaya odaklımamızı sağlayan çevik bir süreç
- Bu yöntem **çalışan yazılımımız** (iki haftadan bir aya kadarlık sürede) hızlı ve verimli üretimine olanak sağlar.
- **İş önceliklerini** belirlerin.
- Takımlar, **en yüksek öncelikli** özelliklerini geliştirmenin en uygun yolunu belirlemek için **kendi kendini yönetir**.
- Her iki haftada ya da ayda bir **çalışan yazılım** parçası üretilir
- Takımlar ürünü geliştirmeye iterasyonlarla devam ederler.

Genel özellikler

- Kendi kendini organize eden takımlar
- Ürün bir dizi "sprint" yinelemeler ile ilerler
- Gereksinimler ürün özellik birikim listesine dönüştürülür
- Çevik bir ortam yaratmak için çevik prensiplere sadık kalınır.
- Scrum en çok kullanılan "Çevik süreçlerden" biridir.

• degilim sprint sırasında
yapılmıyor

Scrum'un Tarihçesi

1995:

- Yeni bir yöntem olarak: Scrum, Jeff Sutherland ve Ken Schwaber
- Scrum Geliştirilmesi ve Extreme Programlama ile Scrum kombinasyonu: Mike Beedle

1996:

- Uluslararası konferansta Scrum tanıtımı

2001:

- "Scrum ile Çevik Yazılım Geliştirme" yayını Ken Schwaber ve Mike Beedle

Birçok şirkette Scrum'un başarılı olarak kullanılmaktadır. Geliştirenler, Agile Alliance üyeleri

Scrum Çerçevesi

- **Roller:** Ürün Sahibi (Product Owner), ScrumMaster, Takım
- **Törenler:** Sprint Planlama, Sprint İncelemesi, Sprint Retrospektifi (geriye bakış) ve Günlük Scrum Toplantısı
- **Eserler:** Ürün İstek Listesi, Sprint Backlog ve Burndown Tablosu

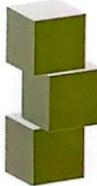
Sprints (Koşu)

- Scrum projeleri bir dizi "sprint" ile ilerleme kaydeder
- XP iterasyon (yinelemelerine) benzer
- Hedef süre bir aydır (Son iki senedir 15 güne kadar düştü)
 - (+/-) bir veya iki hafta
 - Ancak, sabit bir süre daha iyi bir ritme yol açar
- Ürün parçası sprint sırasında tasarlanmış, kodlanmış ve test edilmişdir

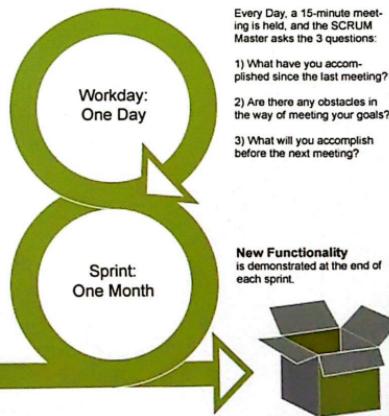
Sprint Iteration

SCRUM Sprint Cycle

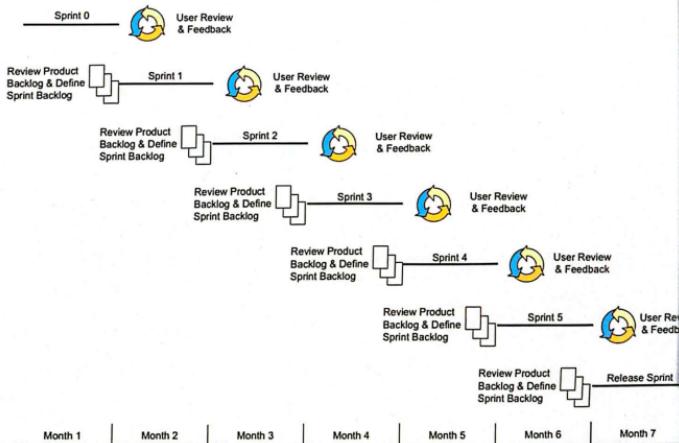
Product Backlog:
Prioritized list of features required by the customer



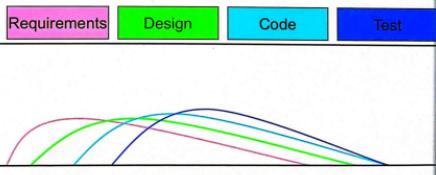
Sprint Backlog:
Features to be done this sprint.
Features are expanded into smaller tasks.



Aylık periodlarda ürüne dönüşen kod parçaları



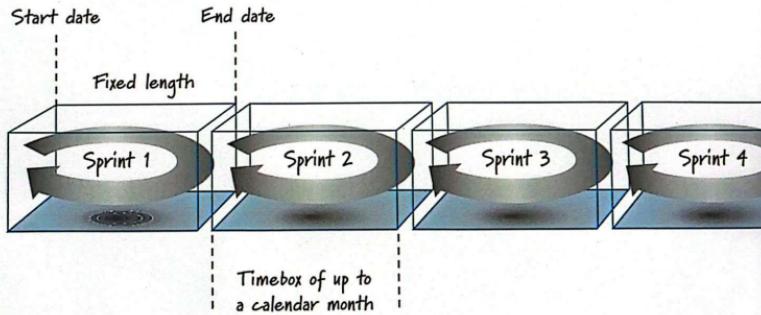
Sequential vs. Overlapping Dev



Waterfall vs Iterative vs Agile

<p>Requirement Analysis</p> <p>Design</p> <p>Coding</p> <p>Testing</p> <p>Implementation</p>	<p>R1 R2 R3</p> <p>D1 D2 D3</p> <p>C1 C2 C3</p> <p>T1 T2 T3</p> <p>Implementation</p>	<p>R1 D1 C1 T1 I1</p> <p>R2 D2 C2 T2 I2</p> <p>R3 D3 C3 T3 I3</p> <p>R4 D4 C4 T4 I4</p> <p>R5 D5 C5 T5 I5</p>
<p>Water Fall Life Cycle Model</p> <p>Each phase is being executed completely until next phase starts.</p>	<p>Iterative Life Cycle Model</p> <p>Each phase is divided into n more phases – 3 in the above diagram. Each phase being executed 3 times. Design starts only after all 3 iterations of requirements are over</p>	<p>Agile Life Cycle Model (Iterative and Incremental)</p> <p>Implementation happens multiple time and incremental. Set R1, D1, C1, T1, I1 is called Scrum 1</p>

Scrum



Product Owner (Ürün sahibi)

- Ürünün özelliklerini tanımlar
- Sürüm tarihi ve içeriğine karar verir
- Ürünün karlılığından (ROI) sorumludur
- Özellikleri piyasa değerine göre önceliklendirir
- Özellikleri ayarlar ve gerektiğinde her koşuda öncelikleri değiştirir
- İş sonuçlarını kabul eder veya ret eder

Scrum Takımı

- Genellikle 5-10 kişi (8 ideal)
- Çapraz fonksiyonel
 - Kalite Güvence, Programcılar, UI Tasarımcıları, vb.
- Üyeler tam zamanlı olmalı
 - İstisnalar olabilir (ör. Sistem Yöneticisi, vb.)
- Takımlar kendi kendilerini organize ederler
 - Bir takım ekibi birinden kendi kendine organize ederse ne yapmalı?
 - Ideal olarak, hiçbir başlık ama nadiren bir olasılık
- Üyelik sadece sprintler arasında değişebilir

Pre-Project/Kickoff Meeting (Proje Öncesi / Başlangıç Toplantısı)

- Sprint Planlama Toplantısının özel bir şekli
- Proje başlangıcından önce yapılan toplantı

Sprint

- Ürün işlevsellliğini artırdığı bir ay (ya da 15 gün) süren iterasyon (yneleme)
- Hiçbir dış etki Sprint boyunca Scrum takımının yapacaklarını etkilemez
- Her Sprint Günlük Scrum Toplantısı ile başlar

Daily Scrum (Günlük Scrum)

- Problem çözme oturumu değildir
- Kimin planın arkasında olduğunu bulmanın bir yolu değildir
- Takım üyelerinin birbirleriyle ve Scrum Master'a yapacakları işlerle ilgili sözler verdikleri bir toplantıdır.
- Takımın ilerlemesini izlemek için Scrum Master için iyi bir yoldur

The Scrum Master

- Proje yönetime temsil eder
- Scrum değerlerini ve uygulamalarını uygulamaktan sorumludur
- Engeller kaldırır (takıma ayakbağı olabilecek)
- Takımın tamamen işlevsel ve urenken olmasını sağlar
- Tüm roller ve işlevler arasında yakın işbirliğini etkileşim sağlar
- Takımı harici müdahalelerden korur

Ceremonies (Törenler)

- Sprint Planlama Toplantısı
- Sprint
- Günlük Scrum Toplantısı
- Sprint İnceleme Toplantısı (Retrospektif)

Sprint Planlama Toplantısı

1. bölüm:

Ürün İş Listesi Oluşturma (Product Backlog)

Sprint Hedefini Belirleme.

Katılımcılar: Ürün Sahibi (Product Owner), Scrum Master, Scrum Takımı

2. bölüm:

Katılımcılar: Scrum Master, Scrum Takımı

Sprint İş Listesi oluşturma (Creating Sprint Backlog)

Daily Scrum (Günlük Scrum)

- Parametreler
 - Günlük
 - 15 dakika
 - Ayakta
 - Problem çözme için yapılmaz
- Üç soru cevaplanır:
 - Dün ne yaptı?
 - Bugün ne yapacaksın?
 - Yolunda hangi engeller var?

Scrum Sıkça Sorulan Sorular

Scrum Ölçeklenebilirliği

Neden günlük?

- "Bir proje bir yıl nasıl gecikir?"
»Her gün birazık.», Fred Brooks, The Mythical Man-Month.
 - Scrum toplantıları e-postayla gönderilen durum raporları ile değiştirilebilir mi?**
 - Yok hayır
 - Bütün takım bütün resmi her gün görüyor olmalı
 - Yapacağınızı söylediğiniz şeyi yapmak için meslektaş baskısı oluşturur

- Tipik bir Scrum takımı 6-10 (8 ideal) kişidir.
 - Jeff Sutherland - 800'den fazla kişi olabilir.
 - "Scrum of Scrums" ya da "Meta-Scrum" denen yapı
 - Toplantıların sıklığı, yazılım paketleri arasındaki kuplej (coupling) derecesine dayanmaktadır.

Çevik Yazılım Hayat Döngüsü



Scrum Avantaj/Dezavantaj

Avantajları

- Kısa iterasyonlarda tamamen geliştirilmiş ve test edilmiş özellikler
 - Sürecin uygulama basitliği
 - Açıkça tanımlanmış kurallar
 - Artan üretkenlik
 - Kendi kendini düzenleme
 - Her takım üyesi fazlasıyla sorumluluk taşıır
 - İyileştirilmiş etkileşim
 - Uç Programlama ile Kombinasyon

Dezavantajları

- Yazılı doküman azlığı
 - Sorumluluk ihlali olasılığı
(iyi niyet üzerine kurulu)

Kısaca Agile

- Çevik = deneyin, inceleyin, uyarlayın, tekrarlayın
- Son derece odaklanmış, güçlendirilmiş ekipler kurmayı hedefler.
- Tüm paydaşlarla işbirliği yapmak zorunluluğu getirir.
- Geri bildirimini optimize et ve otomatikleştirir.
- Müşteriye gerçek değeri erken ve sık sunum imkanı sunar.
- Değerlendirmek, öncelik vermek ve yeniden planlamak için geri bildirimlerin kullanımı sağlar.
- Yüksek kalite sunar, esnekliği sağlar
- Her aktivitenin iş değerini değerlendirebilmesini sağlar.

Çevik Yöntemler ve Süreç Olgunluğu

Farklı Hedefler

– Disiplinli Yöntemler (CMMI süreçleri)

- Tutarlılık
- İstikrar
- Ongörülebilirlik

– Çevik Yöntemler

- Hızlı değişime cevap verin
- İnovasyonu teşvik ediyor
- "WYSIWYG"

Beş Kritik Boyut

