GÜNÜMÜZDE YAZILIM PROJELERİNİN DURUMU

Yazılım geliştirme süreci sıkıntılı ve uzun süren bir dönemdir. Yazılım projeleri yönetimsel eksikliklerden dolayı ancak kısmi başarı ve memnuniyet ile tamamlanabilmektedir. Ülkemizde geliştirilen projelerin başarıya ulaşmasına katkı sağlamak amacıyla, Araştırma Destek Programları Başkanlığı (ARDEB) tarafından desteklenen projelerin çıktı, sonuç ve etkilerini nicelik ve nitelik olarak artırmak amacıyla yüksek başarı ile sonuçlanan projelerin yürütücü ve araştırmacılarını ödüllendirmek için TÜBİTAK tarafından belirlenen ölçütler ve değerlendirme yöntemine göre hesaplanarak, proje ekibine (yürütücü ve araştırmacılara) TÜBİTAK Proje Performans Ödülü (PPÖ), denilen bir teşvik ödülü verilmektedir.

Yazılım Projelerinde Başarısızlığın Nedenleri Aşağıdaki Gibi Listelenebilir;

- Müşterinin isteklerini doğru analiz edememek
- Proje için uygun ekibi kuramamak
- Yanlış teknoloji ve mimari seçimleri
- Geleneksel yöntemlerin eksiklikleri
- Müşteriyle iletişimden kaçınmak vs.

ÇEVİK (AGILE) YAZILIM GELİŞTİRME

Çevik, dünyada yazılım süreçlerini daha esnek ve güçlü kılmak için kullanılan aynı zamanda yazılım süreçlerini de kısaltan kavramsal bir yazılım geliştirme metodolojisidir.

- Bu metodolojide projenin ölçeği ne olursa olsun, proje küçük yinelemelere (iteration) ayrılır ve her yineleme başlı başına bir proje gibi ele alınarak geliştirilir.
- Her yinelemenin sonunda da proje ekibi tarafından müşteriye, projenin ne kadarının gerçekleştirildiğine dair bilgi verilir.
- Çevik ile her yinelemenin 2-4 hafta kadar sürmesi planlanmaktadır.
- Her yinelemenin kendi içerisinde çalışan bir sistem olması sonucu müşteriye sürekli çalışan bir yazılım teslim edilerek, müşteri memnuniyetinin artması sağlanmaktadır.
- Çevik'in hızı proje ekibinde çalışan tüm ekip üyelerinin sürekli iletişim halinde olmasından kaynaklanmaktadır.
- Projenin küçük parçalardan oluşması da geriye dönük hataların düzeltilmesini kolaylaştırmaktadır.

Çevik (Agile) Yazılım Geliştirme Manifestosu

Kent Beck ve değişik çevik metotlarının temsilcileri olan 16 arkadaşı 2001 yılında bir araya gelerek ortak yanlarını ortaya koyan bir çevik manifestosu yayınladılar. Bu manifestoda;

- Süreçler ve Araçlar yerine **Bireysel ve Etkileşimler**,
- Kapsamlı ve Belgeler yerine **Çalışan Yazılım,**

- Sözleşme Görüşmeleri yerine **Müşteri İlişkileri,**
- Plan İzleme yerine **Değişikliğe Uyum Sağlayabilme**

daha önemli ve öncelikli olduğu belirtilmektedir.

Çevik (Agıle) Yazılım Geliştirme Prensipleri

- Müşteriye hızlı ve sürekli olarak kullanılabilir yazılım teslimatı yapmaktır.
- Kodlamanın ilerleyen safhalarında bile gereksinim değişiklikleri kabul edilir, esneklik vardır.
- Mümkün olduğunca kısa zaman aralıklarında çalışan, kaliteli yazılım teslimatı yapılır.
- Analistler, uzmanlar, yazılımcılar, testçiler vs. tüm ekip elemanları birebir iletişim halinde birlikte çalışırlar.
- İyi projeler motivasyonu yüksek kişiler etrafında kurulabilir. Ekibe ihtiyaç duydukları desteği ve ortamı sağlamak ve onlara güvenmek önemlidir.
- Ekip içerisinde kaliteli bilgi akışı için yüz yüze iletişim önemlidir.
- İlerlemenin birincil ölçüsü çalışan bir yazılımdır.
- Çevik süreçler, sürdürülebilir geliştirmeyi destekler.
- Sağlam teknik altyapı ve tasarım çevikliği artırır.
- Basitlik önemlidir.
- En iyi mimariler ve tasarımlar kendini organize edebilen ekipler tarafından yaratılır.
- Düzenli aralıklarla ekip kendi yöntemlerini gözden geçirir ve verimliliği artırmak için gerekli iyileştirmeleri yapar.

Yazılım Projelerinde Yaygın Kullanılan Çevik Metodolojiler

- Extreme Programming (XP),
- SCRUM,
- Kanban,
- Rational Unified Process (RUP),
- Feature-Driven Development (FDD),
- Test-Driven Development (TDD),
- LEAN Development,
- Dynamic System Development Methodology (DSDM)

EXTREME PROGRAMMING (XP)

Extreme Programming (XP), Kent Beck tarafından 1999 yılında bir yazılım geliştirme disiplini olarak ortaya çıkarılmıştır. Yazılım geliştirme süreçlerinde kalite ve verimliliği artırmak amacıyla tasarlanmış bir çevik (agile) metodolojisidir. XP kolay, grup içi iletişime önem veren, geri dönüşlerin daha fazla olmasına imkan sağlayan bir yazılım geliştirme yöntemidir. Özellikle değişken müşteri gereksinimleri ve hızlı teslimat beklentileriyle karşı karşıya kalan projeler için uygundur.

Extreme Programming (XP)'nin Temel Değerleri

- İletişim: Projenin başarıya ulaşabilmesi için öncelikle ekibin sağlıklı iletişim içinde olması gerekir. XP bu iletişim eksikliğini ortadan kaldırmayı amaçlar. İletişim de yüz yüze olmalıdır.
- Basitlik: Basitlik, zorunlu işlerin yapılmasıdır. Basitliği en iyi şekilde sağlamak için, günün ihtiyaçlarını hedef alarak, esnek ve basit bir sistem gerçekleştirmeye çalışır. Karmaşık çözümler XP'nin mantığına aykırıdır.
- Geri Bildirim: Ortaya çıkabilecek hataları en aza indirmek amacıyla kullanılır.
- Cesaret: Projelerin üzerine yılmadan gidilmesi, projelerin geliştirilmesi açısından son derece önemlidir.

Extreme Programming (XP)'nin Pratikleri

- **Planlama Oyunu:** Müşterinin belirlediği her bir yineleme için yazılım ekibinin müşterinin de içinde bulunduğu bir toplantıda, o işin ne kadar zamanda yapılacağıyla ilgili kestirimde bulunmasıdır.
- **Ekipte Müşteri:** İşleri gerçekleştirmek için müşterinin varlığına ihtiyaç duyulmaktadır.
- **Önce Test:** Kod yazılmadan önce test programı yazılır.
- **Basit Tasarım:** Müşterilerin gereksinimlerini karşılayacak en basit tasarım gerçekleştirilir.
- **Çiftli Programlama:** Programın geliştirilme aşamasında her bir programcı sahip olduğu yetenekler ve bilgi birikimleri sayesinde projenin geliştirilmesine yardımcı olur.
- **Sürekli Entegrasyon:** Yazılım geliştirilirken yapılan sistem değişiklikleri ve yeni bileşenler hemen sisteme entegre edilerek günlük derlemelerle test edilir.
- **Kısa Aralıklı Sürümler:** Proje birbirinden ayrı zaman aralıklarına (2-4 hafta) bölünür. Her bir zaman aralığında yapılacak işin kendine ait son teslim tarihi vardır. Belirlenen son teslim tarihini aşmayarak tamamlanır ve müşteriye teslim edilir.
- **Yeniden Yapılandırma:** Kod ve tasarım sürekli gözden geçirilir.
- **Ortak Kod Sahiplenme:** Geliştirilen yazılım kodu, bütün ekip üyelerinin ortak malıdır.

- **Benzetim:** Gerçekleştirilecek yazılımda sistemler birbirine benzetilerek yazılım geliştirmeye çalışılır.
- **Kodlama Standardı:** Ekip üyeleri önceden tanımlanmış kodlama standartlarına göre yazılımı gerçekleştirir. Burada amaç, yazılan kodun karmaşıklığını azaltarak bütün ekip üyeleri tarafından kolaylıkla anlaşabilmesini sağlamaktır.
- **Haftada 40 saat:** Çalışma verimliliği açısından haftada 40 saatlik bir çalışma süresi ayrılır. İşler bu zaman diliminde bitirilir.

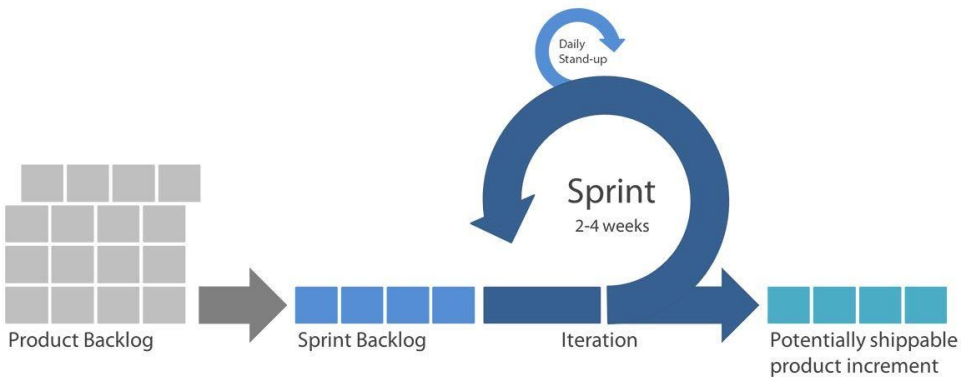
SCRUM

Yazılım projelerini yönetme de kullanılan yinelemeli ve artımlı çevik yazılım geliştirme metodolojisidir. Scrum, Rugby sporundaki bir hücum taktiğinin adıdır. Rugby'de olduğu gibi Scrum'da takım toplanır, planlama oyunu oynar ve görevler dağılarak herkes tek bir hedef için çalışır. Bu taktikte amaç, tüm oyuncularla birlikte topu karşı sahaya taşıyıp atak yapmaktır.

- SCRUM model, gereksinimleri açıkça belirli olmayan, değişime açık ve karmaşık yazılım projelerinin yönetimi için uygulanmaktadır.
- Yazılım projelerinde izlenmesi gereken adımlar detaylı bir şekilde belirtilmemektedir, onun yerine basit ama önemli birkaç olmazsa olmaz kuralıyla esnek bir yönetim sunmaktadır. Karmaşık yazılım işlerini küçük birimlere (sprint) bölerek geliştirmeyi öngörür.
- Bu metodolojide bir yinelemenin tamamlanması 2-4 haftadan fazla sürmemekte ve günlük 15 dakikalık toplantılarla sürekli iş takibi yapılmaktadır.

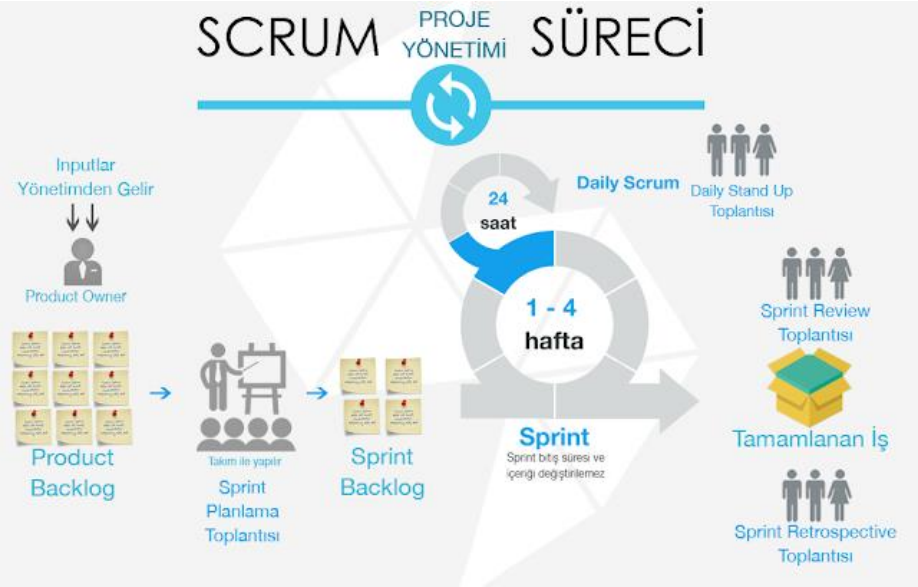
SCRUM'ın genel çalışma mantığı:

- Product Backlog: Müşteri ile anlaşıp önceliklendirilmiş yüksek seviyeli gereksinim listesidir.
- Sprint Backlog: Aktif olarak yapılacak öncelikli gereksinim listesini ifade eder.
- Scrum Daily Meeting: Her gün Scrum takımıyla 15 dakikalık düzenlenen toplantılardır.
- Potentially shippable product increment: Çalışan sürüm müşteriye teslim edilir.



Kısaca,

- Bir Product Owner (ürün sahibi)'ın karmaşık bir sorun için Product Backlog (ürün gereksinim listesi)'daki işleri sıraladığı,
- Scrum Team (scrum takımı)'in, seçilen işleri sprint (periyot) boyunca bir değer önceliklendirdiği,
- Scrum Team ve paydaşların sonuçları inceleyip ve bir sonraki Sprint için düzeltme y aptığı,
- Ve bunları tekrarladığı bir ortamı yaşatmak için bir Scrum Master gereklidir.



Scrum Temel Kavramlar:

Scrum'da üç temel kavram vardır. Bunlar;

- Roller (Rollers)
  - Ürün Sahibi (Product Owner)
  - Scrum Yöneticisi (Scrum Master)
  - Scrum Takımı (Scrum Team)
  - Toplantılar (Meetings)
    - Koşu Planlama (Sprint Planning)
    - Koşu Gözden Geçirme (Sprint Review)
    - Günlük Scrum Toplantısı (Daily Scrum Meetings)
- Bileşenler/Araçlar (Artifacts)
  - Ürün Gereksinim Dokümanı (Product Backlog)
  - Koşu Dokümanı (Sprint Backlog)

- Sprint Kalan Zaman Grafiği (Burndown Chart)

Roller

- Ürün Sahibi (Product Owner): Projenin iş değeri açısından geri dönüşü ile sorumludur. Ekibin bir parçasıdır, müşteri tarafından görevlendirilmiştir, detayları takip eder, geri dönüşler verir.
- Scrum Yöneticisi (Scrum Master): Takımın Scrum'ın temel değerlerine, pratiklerine ve kurallarına bağlı kalmasını garanti altına alır. Takım ve organizasyonu Scrum'a adapte eder.
- Scrum Takımı (Scrum Team): Devamlı iletişim halinde olan ve tek bir hedefe ulaşmak için mücadele eden kişilerden oluşur.

Toplantılar (Meetings)

- Koşu Planlama (Sprint Planning): Product backlog ile belirtilen gereksinimler, bu toplantı ile geliştirme takımı tarafından küçük görevlere (task) ayrılır. Takımdaki her bir kişi kendi hızına göre bu görevleri kendilerine alır. Bu toplantıya product owner, geliştirme takımı ve scrum master katılır.
- Koşu Gözden Geçirme (Sprint Review): Her sprint sonunda yapılır. Yapılan sprint gözden geçirilir, ortaya çıkan ürün değerlendirilir. Amaç yazılımın ürün sahibinin gereksinimlerine uygun olarak geliştirildiğinden emin olmaktır. Eğer bir hata var ise farkedilir ve düzeltilir.
- Günlük Scrum Toplantısı (Daily Scrum Meetings): Her gün aynı yerde aynı saatte ayak üstü yapılan 15 dakikalık toplantılardır. Üyeler davet edilmeyi beklemezler. Bu toplantı gelecek 24 saati planlamak üzere yapılır. Takımdaki her üye dün ne yaptım, bugün ne yapacağım, işimi engelleyen herhangi bir sorun var mı sorularına cevap verir. Bu sayede herhangi bir sorunu var ise scrum master bu problemi ortadan kaldırır.

Bileşenler/Araçlar (Artifacts)

- Ürün Gereksinim Dokümanı (Product Backlog): Proje için gerekli olan gereksinimler listesidir. Proje sonunda "Ne üretilmek isteniyor?" sorusuna cevap aranır. Product owner tarafından müşteriden gereksinimler alınır, öncelik sırasına göre sıralanır. Product owner, değişen ihtiyaçlara göre product backlog'a ekleme veya çıkarma yapabilir. Böylece değişim, projenin her aşamasında projeye kolayca entegre edilebilir olur.
- Koşu Dokümanı (Sprint Backlog): Geliştirme takımı tarafından product backlog item'lar öncelik sırasına göre sprint içerisine alınırlar. Bir sprint boyunca yapılacak item'ların listesini oluşturur. İşlerin detaylı olarak zaman çizelgesi çıkarılır.
- Sprint Kalan Zaman Grafiği (Burndown Chart): Yatay ekseninde sprintin günlerini, dikey ekseninde sprintte kalan işi gösteren grafiktir. Scrum'un temel ilkelerinden olan şeffaflığı sağlar.

Professional Scrum Master (PSM)

- Profesyonel Scrum Master (PSM) sertifikası, Scrum çerçevesi, Scrum Master sorumlulukları ve Scrum'ın nasıl uygulanacağı konusundaki bilginizi doğrular.
- Sertifikalı Scrum Master olmak istiyorsanız dünya genelinde kabul görmüş iki büyük organizasyon üzerinden sınavlarına girerek sertifikanızı alabilirsiniz. İlki scrumalliance.org, diğeri ise scrum.org.
- Scrum kılavuzu: https://scrumguides.org/docs/scrumguide/v1/Scrum-Guide-TR.pdf

SENARYO: Bir e-ticaret web sitesi geliştirmesini scrum modele göre düzenleyiniz?

1. Ürün Sahibi ile Başlangıç

Proje, bir ürün sahibi (Product Owner) tarafından yönetiliyor. Ürün sahibi, kullanıcı ihtiyaçlarına ve iş hedeflerine göre bir ürün listesi (Product Backlog) oluşturur.

2. Kullanıcı Hikayeleri

- Kullanıcı Hikayesi 1: "Bir müşteri olarak, ürünleri kategoriye göre filtrelemek istiyorum."
- Kullanıcı Hikayesi 2: "Bir müşteri olarak, alışveriş sepetimi görmek istiyorum."
- Kullanıcı Hikayesi 3: "Bir müşteri olarak, siparişimi tamamlamak için ödeme yapabilmek istiyorum."

3. Sprint Planlama

Proje ekibi, belirli bir süre (örneğin, 2 hafta) için bir sprint planlar. İlk sprintte, aşağıdaki kullanıcı hikayeleri üzerinde çalışılacak:

Kategoriye göre ürün filtreleme.

Alışveriş sepetinin görüntülenmesi.

4. Geliştirme Süreci

- **Günlük Scrum Toplantıları**: Her gün, ekip üyeleri 15 dakikalık kısa toplantılarda şu üç soruyu yanıtlar:

  Dün ne yaptım? Bugün ne yapacağım? Herhangi bir engel var mı?

- **Sprint Sonrası İnceleme**: Sprint sonunda, ekip geliştirdikleri özellikleri gösterir. Ürün sahibi, kullanıcı hikayelerinin nasıl gerçekleştirildiğini değerlendirir.

5. Geri Bildirim ve İyileştirme

Sprint incelemesinin ardından, ekip bir retrospektif toplantısı yapar. Bu toplantıda, nelerin iyi gittiği, nelerin geliştirilmesi gerektiği ve gelecekteki sprintlerde ne gibi değişiklikler yapılacağı tartışılır.

6. Yeni Sprint Planlama

Elde edilen geri bildirimler doğrultusunda, bir sonraki sprint için yeni kullanıcı hikayeleri belirlenir. Örneğin:

- Ödeme işlem sürecinin geliştirilmesi.
- Kullanıcı hesaplarının yönetilmesi.

7. Sürekli İyileştirme

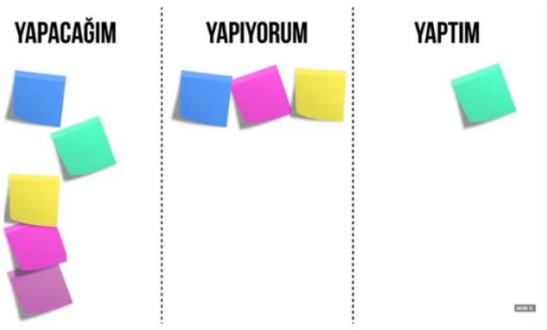
Scrum çerçevesi içinde ekip, her sprintte hem ürün hem de süreç üzerinde sürekli iyileştirme yapar. Kullanıcıların geri bildirimleri doğrultusunda, ürün özellikleri ve geliştirme yöntemleri zamanla optimize edilir.

KANBAN

Kanban, bir süreç içinde hareket ederken işi yönetmek için kullanılan görsel bir sistemdir. Yalın ürün geliştirme, üretim süreçlerinde başarıyla kullanılan yalın metodolojilere dayanmaktadır. Kanban bir agile proje yönetim tekniğidir.

Kanban, hem süreci (iş akışını) hem de bu süreçten geçen fiili işi görselleştirir. Kanban'ın amacı, sürecinizdeki olası darboğazları belirlemek ve bunları düzeltmektir, böylece iş optimum hızda veya verimde gerçekleşebilir. Kanban iş akışını optimize etmenize ve ekibinizin tam kapasitesini kullanmanıza yardımcı olmak için tasarlanmış bir yöntemdir.

Kanban Metodolojisinin Pratikleri



- İş akışını görselleştirin: Mevcut iş akışını anlayın, bir öğeyi talepten teslim edilebilir bir ürüne taşımak için uygulanacak adımların sırası nedir?
- Devam Eden Çalışmayı Sınırla: Odaklanma kaybı, ekip performansına zarar verebilir, bu uygulama, devam eden işe sınırlar koyarak kesintileri ortadan kaldırmaya odaklanır. Devam eden çalışma için sınırlar uygulayan ekipler, yeni işe başlamadan önce olağanüstü işleri bitirmeye odaklanır.
- Akışı yönetin: Akış verimliliğini gözlemleyerek ve analiz ederek, sorunlu alanları tanımlayabilirsiniz. Teslim sürelerini iyileştirerek ve gecikmeleri önleyerek sorunsuz bir iş akışı oluştur.

- Süreç politikalarını açık hale getirin: Süreç, ekipteki herkes için açıkça tanımlanmalı ve onaylanmalıdır.
- Geri bildirim döngülerini kullanın: Olumlu değişimler ve geribildirimler için düzenli toplantılar gereklidir. Bu toplantıların sıklığı değişir, ancak fikir düzenli olarak belirli bir zamanda olmalarıdır.
- İşbirliği yaparak geliştirin: Takımlar süreç hakkında ortak bir anlayışa sahip olduklarında, değişimlerde bir sorun çıkması durumunda bir fikir birliğine varma olasılıkları yüksektir.

Kanban Uygulamanın Faydaları Nelerdir?

- **İşleri yönetmek için daha şeffaf bir sistem sunar**

Kanban, işleri görsel olarak takip edebilmenizi sağlar. Bu sayede, işlerin ne zaman başladığını, ne kadar sürede tamamlanacağını ve işlerin hangi aşamada olduğunu görebilirsiniz. Bu, tüm çalışanların bir işin durumunu anlamalarını ve işin takibini kolaylaştırır.

- **İşleri önceliklendirmenize yardımcı olur**

Kanban kartları, işlerin önceliklerini belirlemenize yardımcı olur. Öncelikli işlerinizin kartları daha önde dururken, diğer işlerin kartları geri planda kalır. Bu sayede, her zaman öncelikli olan işleri takip edebilirsiniz.

- **İşlerin tamamlanma süresini azaltır**

Kanban, işlerin tamamlanma süresini azaltmanıza yardımcı olur. Bu, işlerin hangi aşamada olduğunu anlamanıza ve işleri hızlandırmak için gerekli önlemleri almanıza olanak tanır. Müşteri memnuniyetini artırır ve organizasyonun daha fazla iş yapmasına yardımcı olur.

- **Kaynak kullanımını optimize eder**

Kanban, işlerin her aşamasında ne kadar kaynağa ihtiyaç duyulduğunu belirlemenizi sağlar. Bu sayede, kaynaklar optimum şekilde kullanılır ve gereksiz maliyetlerden kaçınılabilir.

- **Sürekli iyileştirme sağlar**

Kanban, işlerin her aşamasında daha verimli olmanızı sağlar. İşleri tamamlama süresini azaltmak ve kaynak kullanımını optimize etmek için sürekli olarak iyileştirmeler yaratır.

- **Takım çalışmasını teşvik eder**

Kanban, çalışanlarının işleri daha iyi takip etmelerine yardımcı olur ve birbirleriyle daha iyi işbirliği yapmalarını sağlar. Bu, takım çalışmasını teşvik eder ve işletmenin başarısını artırır.

RATIONAL UNIFIED PROCESS (RUP)

Nesne yönelimli modeller için bir yazılım geliştirme sürecidir. Program geliştirmenin tüm aşamaları ve yönleri için kılavuzlar, örnekler ve şablonlar sağlayan çevrimiçi bir mentor olarak hizmet eder. Bir projenin veya yazılımın geliştirilmesi, her aşamada çeşitli faaliyetlerin

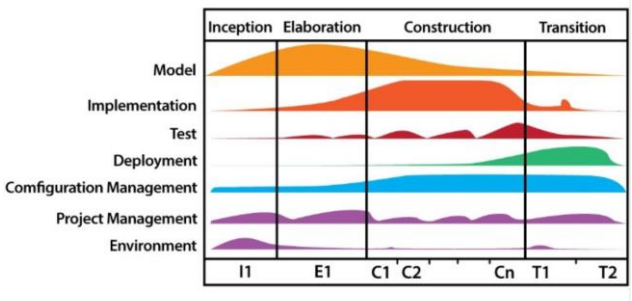
gerçekleştirildiği dört aşamaya ayrılır. Sürecin temel faaliyetleri proje boyunca tekrar ettiği için RUP yinelemeli veya tekrarlı olarak kabul edilir. Bileşenleri ayarlanabilir ve döngü aşamaları proje veya yazılım gereksinimlerini karşılayana kadar tekrarlanabilir, bu da onu çevik hale getirir.

RUP, yazılım geliştirme yaşam döngüsünü birbirini takip eden dört aşamaya ayırır:

- Başlangıç (Inception)
- Detaylandırma (Elaboration)
- Yapım (Construction)
- Geçiş (Transtion)

Her aşama, sistematik ilerleme için bu yinelemelerde ulaşılması gereken belirli hedeflerle birlikte birden fazla yinelemeden oluşur. Bu aşamalarda derinlemesine analiz, geliştirme, test ve entegrasyon faaliyetleri gerçekleştirilerek yazılım projelerinin etkin yönetimi sağlanır.

- **Başlangıç (Inception):** Projenin başlangıç kapsamının anlaşılmasına, hedeflerinin tanımlanmasına ve sistemin uygulanabilirliğinin oluşturulmasına odaklanır. Geliştirme ekibi, üst düzey gereksinimleri belirlemek ve yakalamak, riskleri belirlemek ve bir başlangıç proje planı ve maliyet tahminleri geliştirmek için paydaşlarla işbirliği yapar.
- **Detaylandırma (Elaboration):** Geliştirme ekibi, başlangıç aşamasından gelen geri bildirimleri birleştirerek sistemin mimarisini ve gereksinimlerini geliştirir. Bu aşama, sistem gereksinimlerinin daha ayrıntılı bir analizini, kullanım senaryosu modellerinin oluşturulmasını ve potansiyel risklerin ve azaltma stratejilerinin tanımlanmasını içerir. Detaylandırma aşamasının temel amacı, proje için istikrarlı bir mimari ve rafine bir plan oluşturmak, böylece belirsizliği ve ilerleme riskini azaltmaktır.
- **Yapım (Construction):** Sistemin geliştirilmesinin büyük kısmının gerçekleştiği aşamadır. Burada geliştirme ekibi, yazılım bileşenlerini oluşturur, aşamalı olarak işlevsellik ekler ve yinelemeli süreç boyunca sürekli entegrasyon ve testler gerçekleştirir. Belgelerin güncellenmesi ve paydaş geri bildirimlerinin dahil edilmesi bu aşamada önemli görevlerdir. Yapım aşaması, özellikleri tamamlanmış, kapsamlı bir şekilde test edilmiş ve dağıtıma hazır bir sistem uygulamasıyla sona erer.
- **Geçiş (Transtion):** Tamamlanan sistemin dağıtımına ve kullanıcı topluluğuna sorunsuz bir geçiş sağlanmasına odaklanır. Bu, kullanıcıların eğitilmesini, sorunların ele alınmasını ve sistem performansının belirlenmiş başarı kriterlerine göre doğrulanmasını içerir.



Rational Unified Process, yinelemeli geliştirmeyi, risk yönetimini ve etkili proje organizasyonunu destekleyen, yaygın olarak kabul edilen bir yazılım geliştirme metodolojisidir. Artan ilerlemeyi, erken doğrulamayı ve

kullanıcılardan ve paydaşlardan gelen sürekli geri bildirimi vurgulayarak sistem mühendisliğine yapılandırılmış bir yaklaşım sunar.

SENARYO: Bir online eğitim platformu geliştirmek, kullanıcıların kurslara kaydolmasını, içeriklere erişmesini ve etkileşimde bulunmasını sağlaması talep edilmektedir. Bu projeyi Rational Unified Process (RUP) modele göre düzenleyiniz?

1. İhtiyaç Analizi (Inception Phase)

Amaç: Kullanıcıların farklı eğitim içeriklerine erişebilmesi ve öğretim görevlileri ile etkileşimde bulunabilmesi.

Kullanıcı Hikayeleri:

"Bir öğrenci olarak, farklı kategorilerde kurs aramak istiyorum."

"Bir eğitmen olarak, kurs içeriklerini ekleyip güncellemek istiyorum."

"Bir öğrenci olarak, kurs tamamlama sertifikası almak istiyorum."

2. Proje Planlaması (Elaboration Phase)

- **Risk Değerlendirmesi**: Projenin teknik zorlukları, kullanıcı gereksinimlerinin belirsizliği gibi riskler belirlenir.
- **Sistem Mimarisi Tasarımı**: Yazılım mimarisi, kullanıcı arayüzü, veri tabanı tasarımı ve API yapısı oluşturulur.
- **Prototip Geliştirme**: Temel özelliklerin prototipi oluşturulur ve kullanıcı geri bildirimleri toplanır.

3. Geliştirme (Construction Phase)

- **Iterasyonlar**: Proje birkaç iterasyona bölünerek geliştirilir. Her iterasyon belirli özellikleri geliştirmek için kullanılır.

Iterasyon 1: Kullanıcı kaydı ve giriş sistemi.

Iterasyon 2: Kurs listeleme ve arama işlevi.

Iterasyon 3: Kurs içeriklerinin eklenmesi ve güncellenmesi.

Iterasyon 4: Sertifika alma özelliği.

- **Test Süreci**: Her iterasyonda, geliştirilen özellikler için birim testleri ve entegrasyon testleri yapılır.

4. Geçiş (Transition Phase)

- **Kullanıcı Eğitimi**: Platformun kullanımı için öğrencilere ve eğitmenlere eğitim verilir.
- **Pilot Uygulama**: Yazılım, belirli bir kullanıcı grubuyla pilot olarak uygulanır ve geri bildirimler toplanır.

- **Son Düzenlemeler**: Kullanıcı geri bildirimlerine göre gerekli düzeltmeler yapılır.

5. Yayınlama

- **Sistem Yayını**: Online eğitim platformu, tüm kullanıcıların erişimine açılır.
- **Destek ve Bakım**: Yazılımın sürekli desteği sağlanır ve kullanıcıların ihtiyaçları doğrultusunda iyileştirmeler yapılır.

FEATURE-DRIVEN DEVELOPMENT (FDD – ÖZELLİK GÜDÜMLÜ GELİŞTİRME)

FDD, Avustralyalı Jeff De Luca tarafından geliştirilmiş ve Singapur projesinde beraber çalıştıkları Peter Coad tarafından modifiye edilmiştir. Singapur projesi 50 kişi ile 15 ayda tamamlanan bir proje olmuştur. Daha sonra FDD'nin uygulandığı başka bir proje ise 250 kişi ile 18 ayda tamamlanmıştır. FDD değişik boyutlara büyüyebilen, tekrarlanabilir bir süreçtir.

FDD süreci beş ana basamaktan oluşur; her bir basamak için çıkış şartları tanımlanır ve bunlara uyulur.

- Genel sistemin modelinin geliştirilmesi,
- Özellik listesinin oluşturulması,
- Özellik güdümlü bir planlama yapılması,
- Özellik güdümlü bir tasarımın oluşturulması,
- Özellik güdümlü geliştirmenin yapılması

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

TEST-DRIVEN DEVELOPMENT (TDD - TEST GÜDÜMLÜ GELİŞTİRME)

Çevik yazılım geliştirme süreçlerinde önemi her geçen gün daha iyi anlaşılan bir programlama tekniğidir. Projeler büyüdükçe ve projede daha fazla yazılımcı dahil oldukça yazılımın kırılganlığının artması ve yazılım geliştirme süreçlerinin yönetiminin zorlaşması daha etkin ve otomatize teknikler kullanmayı zorunlu hale getirdi. TDD yöntemi ile test süreçleri yazılım geliştirme süreçleri ile entegre yürütüldüğü için yazılım kalitesi artar, süreç yönetim maliyetleri azalır, hata geri dönüş oranları azalır.

TDD'nin temel prensipleri şunlardır:

Test Yazma: İlk adım, geliştirilmesi planlanan bir özellik için bir test yazmaktır. Bu test, özelliğin beklenen davranışını tanımlar.

Testi Çalıştırma: Yazılan test henüz geçerli olmayacak şekilde tasarlandığı için, ilk çalıştırmada test başarısız olur. Bu aşama, testin doğru bir şekilde yazıldığını gösterir.

Kod Yazma: Başarısız olan testi geçmek için gerekli minimum kodu yazarsınız. Bu adımda sadece testi geçmek için gereken en basit çözümü uygularsınız.

Testi Yeniden Çalıştırma: Yeni kodu çalıştırarak testi tekrar test edersiniz. Eğer test başarılıysa, yazdığınız kodun doğru olduğunu kanıtlamış olursunuz.

Refaktörleme: Kodunuzu temizlemek ve optimize etmek için gerekli düzenlemeleri yaparsınız. Bu aşamada testlerin geçerliliğini koruyarak kodu daha okunabilir ve sürdürülebilir hale getirirsiniz.

Tekrar: Bu döngüyü yeni özellikler eklemek veya mevcut özellikleri geliştirmek için tekrarlarsınız.

TDD, yazılım kalitesini artırır, hata oranını düşürür ve geliştiriciye daha fazla güven sağlar. Aynı zamanda, geliştirme sürecinde daha fazla düşünmeyi teşvik eder.

Örnek Senaryo; Günlük hayatta gelen işler çoğunlukla acildir. Bu aciliyet ve yoğunluktan dolayı, işleri hakkıyla yazmak yerine sadece gereksinimi karşılamak üzere hızlıca yazar ve devreye alırız. Ancak bir süre sonra bu özensiz yazılmış kodlar uygulamayı tıkar ve bir bakmışsınız, uygulama genişleyemez hale gelmiş. Ya da canlıya bir haftada çıktığınız bir ürün için iki hafta canlıya hata desteği vermek zorunda kalıyorsunuz. Bunların çoğu da yazılım hatası ya da eksikliğidir.

Yazılımı geliştirirken genellikle mutlu senaryo (happy-path) dediğimiz senaryolara odaklanıyoruz. Ancak ürün canlıya çıktığında bir süre edge case (uç senaryo) ile karşılaşıyoruz. Çoğu durumda da validasyonlar eksiktir. Teknik borçlar boğazımıza kadar dayanmadan aksiyon almamız gerekir.

Örnek senaryolar, tahmin edilenden çok daha yaygın yaşanan durumlardır. Çözmenin ise tek ve basit bir yolu vardır: İlgili özellik (feature) üzerine daha fazla düşünmek. Kod penceresini ilk değil, en son açmak gerekir. Yazacağımız ürünle düşünce aşamasında daha fazla zaman geçirmeliyiz. İşte TDD (Test Driven Development) yaklaşımı tam olarak bunu yapabilmemiz için bize alan açar.

Testleri düşünürken genellikle uç senaryoları göz önünde bulundururuz. Kodu yazarken, bu kodu nasıl yazacağımızı ve gereksinimi nasıl karşılayacağımızı düşünürken, test yazarken de bu özellikte olması gerekenlerle birlikte olmaması gerekenleri, kodu patlatabilecek senaryoları da düşünmeliyiz. İşin güzel tarafı, bu yaklaşımın zorlama değil, içgüdüsel bir yöntem olmasıdır.

LEAN DEVELOPMENT

Verimlilik temasına odaklanmış yani israfı en aza indirerek müşteri değerini en üst düzeye çıkarmayı teşvik eden bir proje yönetimi metodolojisidir. Lean, daha azla daha fazlasını yapmayı hedefleyen bir yaklaşımdır.

Lean yaklaşımının 3M kuralı, israfı ve değer yaratmayan unsurları tanımlamak için kullanılan bir çerçevedir. Muda (İsraf): Değer yaratmayan her türlü aktivite veya süreç. Muda, kaynakları boşa harcayan her şeyi kapsar. Örneğin, gereksiz işlemler, fazla beklemeler, hatalı ürünlerin yeniden işlenmesi gibi durumlar.

Muri (Aşırı Yük): Ekiplerin veya sistemlerin aşırı yüklenmesi durumunu ifade eder. Muri, iş yükünün veya taleplerin makul sınırları aşmasıyla ortaya çıkar. Bu durum, çalışanların stres seviyesini artırır ve hata yapma olasılığını yükseltir.

Mura (Düzensizlik): İş süreçlerinde düzensizlik veya değişkenlik. Mura, süreçlerin tutarsızlığına işaret eder ve bu da verimliliği olumsuz etkiler. Örneğin, belirli bir işin farklı zamanlarda farklı hızlarda yapılması.

DYNAMIC SYSTEM DEVELOPMENT METHODOLOGY (DSDM)

Dynamic System Development Methodology (DSDM), agile yöntemler arasında yer alan ve özellikle yazılım geliştirme süreçlerinde esnekliği ve hızlandırmayı hedefleyen bir çerçevedir. 1990'ların ortalarında geliştirilen DSDM, kullanıcı katılımını teşvik eder ve sürekli geri bildirim ile müşteri ihtiyaçlarına hızlı yanıt verme ilkesine dayanır.

DSDM, aşağıdaki aşamalardan oluşur:

Ön Proje Aşaması: Proje hedefleri ve gereksinimleri belirlenir.

Fizibilite Aşaması: Projenin gerçekleştirilmesi için gerekli kaynaklar ve teknik olanaklar değerlendirilir.

Geliştirme Aşaması: Ürün, iteratif bir şekilde geliştirilir ve kullanıcı geri bildirimleri alınarak sürekli iyileştirilir.

Uygulama Aşaması: Ürün, kullanıcılar tarafından kullanılmak üzere yayına alınır.

Sonuçlandırma Aşaması: Proje sonuçları değerlendirilir ve elde edilen dersler çıkarılır.

ÇEVİK YAZILIM GELİŞTİRME METODOLOJİLERİ İÇİN ÇEŞİTLİ ONLİNE PLATFORMLAR VE ARAÇLAR MEVCUTTUR. BUNLARDAN BAZILARI;

Jira, Atlassian tarafından geliştirilmiş bir proje yönetimi ve takip aracıdır. Özellikle yazılım geliştirme ekipleri arasında yaygın olarak kullanılmaktadır. Jira, çevik (Agile) metodolojilerle uyumlu özellikler sunarak ekiplerin projelerini planlamalarına, izlemelerine ve yönetmelerine yardımcı olur.

Trello, kullanıcıların projelerini ve görevlerini görsel bir şekilde yönetmelerini sağlayan bir proje yönetim aracıdır. Atlassian tarafından geliştirilmiştir ve özellikle basitliği ve esnekliği ile dikkat çeker. Mobil ve web uyumludur.

Asana, projeleri ve görevleri yönetmek için kullanılan bir iş yönetim aracıdır. Ekiplerin birlikte çalışmasını, iş akışlarını düzenlemesini ve hedeflere ulaşmasını kolaylaştıran birçok özellik sunar. Birçok popüler uygulama ve hizmetle entegrasyon sunar (Slack, Google Drive, Microsoft Teams vb.), böylece iş akışlarını daha da optimize edebilir. Mobil uygulaması sayesinde kullanıcılar, projelerine ve görevlerine her yerden erişebilir.

Monday.com, projeleri ve iş akışlarını yönetmek için kullanılan bir iş yönetim platformudur. Kullanıcıların görevleri planlamasına, izlemelerine ve ekip içindeki işbirliğini artırmasına yardımcı olan çeşitli özellikler sunar.

Slack, ekiplerin iletişim kurmasını ve işbirliği yapmasını sağlayan bir anlık mesajlaşma platformudur. Özellikle iş ortamında kullanılır ve çeşitli özellikleri sayesinde ekiplerin daha verimli çalışmasına yardımcı olur.

GitHub, yazılım geliştirme projelerini yönetmek ve sürüm kontrolü sağlamak için kullanılan bir platformdur. Git versiyon kontrol sistemi üzerine kurulmuştur ve özellikle geliştiricilerin işbirliği yapmasına olanak tanır.