İSTANBUL MEDENİYET ÜNİVERSİTESİ

2025-2026 AKADEMİK YILI GÜZ DÖNEMİ

BİL403 YAZILIM MÜHENDİSLİĞİ

4. DERS

Dr. Öğr. Üyesi Ertürk ERDAĞI
erturk.erdagi@medeniyet.edu.tr

Agile (Çevik) Yöntem

Çevik Yaklaşımın Ortaya Çıkışı

- Yazılım mühendisliği tarihinde 1990'ların sonlarına kadar yaygın olarak **plan odaklı (Waterfall)** modeller kullanılıyordu. Ancak bu modellerin en büyük eksikliği, **değişime karşı katı olmalarıydı**. Projeler aylarca hatta yıllarca sürüyor, gereksinimler tamamlandığında kullanıcı ihtiyaçları değişmiş oluyordu.
- Birçok proje, tamamlandığında artık işe yaramayan, pahalı ve kullanıcıyı tatmin etmeyen yazılımlar üretiyordu. Bu durum yazılım sektöründe "**yeni bir felsefe ihtiyacını**" doğurdu.
- 2001 yılında 17 deneyimli yazılım geliştirici ABD'de bir araya gelerek, klasik yöntemlerin eksiklerini tartıştı ve "**Çevik Manifesto (Agile Manifesto)**" adını verdikleri bir belge yayımladı. Bu manifesto, günümüzde çevik yaklaşımların temelini oluşturur.

Agile Manifesto

- Agile Manifesto, yazılım geliştirmede **dört temel değeri** vurgular:
  - **Bireyler ve etkileşimler**, süreçler ve araçlardan daha değerlidir. Başarılı yazılım, insan faktörüne dayanır. İletişim güçlü değilse süreç mükemmel olsa da proje başarısız olur.
  - **Çalışan yazılım**, kapsamlı dokümantasyondan daha değerlidir. Kullanıcıyı etkileyen şey doküman değil, çalışan bir üründür.
  - **Müşteri iş birliği**, sözleşme pazarlığından daha değerlidir. Geliştirme süreci boyunca müşteri aktif rol oynar, sadece başta değil.
  - **Değişime tepki vermek**, plana uymaktan daha değerlidir. Plan önemlidir ama değişime uyum sağlamak başarının anahtarıdır.

- Bu dört değeri destekleyen **12 prensip** vardır.

Agile Manifesto Prensipleri

1. En önemli önceliğimiz değerli yazılımın erken ve devamlı teslimini sağlayarak müşterileri memnun etmektir.
2. Değişen gereksinimler yazılım sürecinin son aşamalarında bile kabul edilmelidir. Çevik süreçler değişimi müşterinin rekabet avantajı için kullanır.
3. Çalışan yazılım, tercihen kısa zaman aralıkları belirlenerek birkaç haftada ya da birkaç ayda bir düzenli olarak müşteriye sunulmalıdır.
4. İş süreçlerinin sahipleri ve yazılımcılar proje boyunca her gün birlikte çalışmalıdırlar.
5. Projelerin temelinde motive olmuş bireyler yer almalıdır. Onlara ihtiyaçları olan ortam ve destek sağlanmalı, işi başaracakları konusunda güven duyulmalıdır.
6. Bir yazılım takımında bilgi alışverişinin en verimli ve etkin yöntemi yüzyüze iletişimdir.
7. Çalışan yazılım ilerlemenin birincil ölçüsüdür.
8. Çevik süreçler sürdürülebilir geliştirmeyi teşvik etmektedir. Sponsorlar, yazılımcılar ve kullanıcılar sabit çalışma temposunu sürekli devam ettirebilmelidir.
9. Teknik mükemmeliyet ve iyi tasarım konusundaki sürekli özen çevikliği artırır.
10. Sadelik, işin özü olmayan işlerin yapılmamasını en üst seviye tutmak elzemdir.
11. En iyi mimariler, gereksinimler ve tasarımlar kendi kendini örgütleyen takımlardan ortaya çıkar.
12. Takım, düzenli aralıklarla nasıl daha etkili ve verimli olabileceğinin üzerinde düşünür ve davranışlarını buna göre ayarlar ve düzenler.

Agile (Çevik) Yaklaşımın Felsefesi

- Agile, bir yöntemden öte bir **zihniyet dönüşümüdür**.
- Amaç, hızlı ve sürekli değer üretmektir.
- Bir Agile takımında hiyerarşi minimumdur.
- Ekip kendi kararlarını verir, kısa döngülerle (sprintlerle) çalışan yazılım parçaları üretir.
- **Agile kültürünün temel unsurları:**
  - Güven, açıklık ve iletişim
  - Hızlı geri bildirim
  - Kendi kendini yöneten ekipler
  - Değişimi kucaklamak
  - Sürekli iyileştirme

Tartışma Sorusu

- Bir yazılım projesinde gereksinimler sürekli değişiyorsa, sizce klasik şelale modeli neden yetersiz kalır?

Tartışma Sorusu

- Bir yazılım projesinde gereksinimler sürekli değişiyorsa, sizce klasik şelale modeli neden yetersiz kalır?
  - **Şelale modeli (Waterfall Model)**, yazılım geliştirme sürecini sıralı adımlar hâlinde ilerleten bir yöntemdir:
    - Gereksinim → Tasarım → Kodlama → Test → Dağıtım → Bakım
    - Her aşama tamamlanmadan bir sonrakine geçilmez. Bu yapı, **değişmeyen, sabit gereksinimler** varsayımıyla çalışır.
  - Ancak **modern yazılım projelerinde gereksinimler dinamik**tir:
    - Müşteri ihtiyaçları proje süresince değişir,
    - Pazar koşulları farklılaşır,
    - Yeni teknolojiler ortaya çıkar,
    - Kullanıcı geri bildirimleri yeni beklentiler doğurur.
  - Şelale modelinde bu değişiklikler sürecin **çok geç aşamalarında fark edilir**.
  - Bir gereksinimin değişmesi, önceki aşamalara dönmeyi gerektirir ki bu:
    - Maliyetli,
    - Zaman kaybı yaratan,
    - Takımı demotive eden bir durumdur.
    - Dolayısıyla şelale modeli, **değişime kapalı** yapısı nedeniyle sürekli evrilen gereksinimleri karşılayamaz.

Agile Yaklaşımları

- Agile yaklaşımının altında farklı çerçeveler (framework) bulunur.
  En bilinenleri:
  - **Scrum**
  - **Kanban**
  - **Extreme Programming (XP)**
  - **Lean Software Development**
  - **Crystal, DSDM, FDD**
- Bu çerçeveler, Agile prensipleri farklı biçimlerde uygularlar.

Agile Yaklaşımları - Scrum

- Scrum, karmaşık ürün geliştirme süreçlerini yönetmek için geliştirilmiş, **iteratif ve artımlı** bir yöntemdir.
- Çalışma kısa döngüler (sprint) halinde yürütülür.
- Her sprint sonunda çalışan bir ürün ortaya çıkar.
- **Scrum Rolleri**
  - **Product Owner (Ürün Sahibi):**
    - Ürünün vizyonunu belirler.
    - Gereksinimleri "product backlog" içinde önceliklendirir.
    - Takımın yaptığı işin değer üretmesini sağlar.
  - **Scrum Master:**
    - Scrum süreçlerinin doğru yürütülmesini sağlar.
    - Engelleri kaldırır, ekibin verimli çalışmasını destekler.
    - Koç, kolaylaştırıcı ve mentordur.
  - **Development Team (Geliştirme Ekibi):**
    - Çok disiplinli bir ekiptir (analiz, tasarım, kodlama, test).
    - Her sprint sonunda çalışan bir yazılım sunar.

Tartışma Sorusu

- Product Owner ile klasik proje yöneticisi arasındaki fark nedir?

Tartışma Sorusu

- Product Owner ile klasik proje yöneticisi arasındaki fark nedir?

Karşılaştırma Alanı	Product Owner (Scrum)	Klasik Proje Yöneticisi (Waterfall / PM)
Temel Odak	Ürünün **değeri ve vizyonu**	Projenin **zaman, maliyet, kapsam** üçlüsü
Amaç	Kullanıcıya en yüksek değeri sunmak	Projeyi zamanında, bütçede ve kapsamda bitirmek
Yetki Alanı	**Ne** geliştirileceğine karar verir	**Nasıl** geliştirileceğini planlar ve yönetir
Takım İlişkisi	Ekip ile eşit, birlikte çalışan bir roldedir	Ekip üzerinde hiyerarşik otoriteye sahiptir
Yönetim Yaklaşımı	Vizyoner, müşteri odaklı	Kontrol ve plan odaklı
İletişim Tarzı	Takım ve paydaşlar arasında köprü	Genellikle yukarıdan aşağı iletişim (raporlama)
Sorumluluk Alanı	Product Backlog, önceliklendirme, kullanıcı değeri	Zaman çizelgesi, bütçe, kaynak yönetimi
Başarı Kriteri	Kullanıcı memnuniyeti, ürünün değeri	Projenin zamanında ve bütçede tamamlanması

Product Owner

•Agile ekibinde **ürünün "ne" olacağına** karar verir.
•**Ürün vizyonunu** tanımlar, hedefleri belirler, müşteri beklentilerini backlog'a dönüştürür.
•Product Backlog'daki iş maddelerini (user story'leri) **önceliklendirir.**
•Takımla birlikte çalışır ama onlara "nasıl yapacaklarını" söylemez.
•Asıl ölçütü: **kullanıcıya ve işletmeye değer katmak.**

Klasik Proje Yöneticisi

Odak noktası **proje yönetim üçgenidir:** zaman, maliyet, kapsam.
Planlama, kaynak tahsisi, risk yönetimi, ilerleme raporlaması gibi görevleri vardır.
Takım üyelerine görev atar, performansı izler.
Genellikle **üst yönetim ve müşteri arasında "raporlama" köprüsüdür.**

Scrum Süreci (Sprint Döngüsü)

- Scrum, birbirini takip eden sprintlerle ilerler. Her sprintin sonunda çalışan bir yazılım teslim edilir.
- **Sprint Planning (Planlama):** Takım, Product Backlog'daki maddelerden o sprintte yapılacak işleri seçer.
- **Sprint Execution (Uygulama):** Takım sprint hedeflerine ulaşmak için çalışır. Günlük toplantılar yapılır.
- **Daily Scrum (Günlük Toplantı):** 15 dakikalık kısa toplantıdır. Her üye:
  - Dün ne yaptım?
  - Bugün ne yapacağım?
  - Engelim var mı? sorularını yanıtlar.
- **Sprint Review:** Sprint sonunda tamamlanan işler müşteriye gösterilir.
- **Sprint Retrospective:** Takım, bir sonraki sprintte süreci nasıl iyileştirebileceğini tartışır.

Scrum Dokümanları (Scrum Artifacts)

- Scrum'da "**artifact**" terimi, sürecin **şeffaflığını**, **ilerlemenin izlenebilirliğini** ve **değer üretimini** sağlayan **belgeleri / araçları** ifade eder. Bu dokümanlar, Scrum ekibinin ne yaptığını, nereye gittiğini ve ne kadar ilerlediğini açıkça gösterir.
- Scrum'da üç temel doküman vardır:
  - **Product Backlog (Ürün Listesi)**
  - **Sprint Backlog (Sprint Listesi)**
  - **Increment (Artırım / Çalışan Ürün Parçası)**

Scrum Dokümanları - Product Backlog (Ürün Listesi)

- Product Backlog, ürünle ilgili **tüm gereksinimlerin, özelliklerin, geliştirme fikirlerinin** yer aldığı, sürekli güncellenen öncelikli listedir. Ürünün vizyonunu yansıtır; yapılacak tüm işler bu listeden seçilir.
- **Sorumlusu:**
  - **Product Owner** — Backlog'un tek sahibidir. Maddeleri belirler, önceliklendirir, güncel tutar.
- **Özellikleri:**
  - Dinamiktir (gereksinimler değiştikçe evrilir).
  - Her maddeye **"User Story"** formatında yer verilir:
  - "Bir kullanıcı olarak [şunu yapmak] istiyorum, çünkü [buna ihtiyacım var]."
  - Maddeler **öncelik sırasına** göre dizilir: en önemli işler en üstte.
  - Geliştirme ekibi, her sprint başında buradan seçim yapar.
- Örnek (Online Eğitim Platformu – Product Backlog)

Öncelik	User Story	Kabul Kriteri
1	Bir öğrenci olarak sisteme kayıt olmak istiyorum	Form doldığunda kayıt başarıyla tamamlanmalı
2	Bir öğretmen olarak ders oluşturmak istiyorum	Ders başlığı ve açıklaması kaydedilmeli
3	Bir kullanıcı olarak şifremi sıfırlayabilmek istiyorum	E-posta üzerinden doğrulama yapılmalı

Scrum Dokümanları - Sprint Backlog (Sprint Listesi)

- Sprint Backlog, bir sprint boyunca yapılması planlanan işlerin listesidir. Product Backlog'daki maddelerin sprint hedeflerine göre seçilip, parçalara ayrılmasıyla oluşturulur. **Sorumlusu: Geliştirme Takımı (Development Team)** — bu listeyi oluşturur ve yönetir. Product Owner rehberlik eder ama sprint backlog'un sahibi takımın kendisidir.
- **Özellikleri:**
  - Sprint planlama toplantısında oluşturulur.
  - Her madde, "görev" düzeyine indirgenir.
  - Sprint boyunca takım kendi planını yönetir.
  - Takım, günlük olarak "ne tamamlandı, ne kaldı"yı buradan izler.
- **Örnek**

Görev	Tahmini Süre	Durum
Kullanıcı kayıt ekranı oluştur	5 saat	Yapılıyor
Veritabanı bağlantısı kur	8 saat	Tamamlandı
Şifre sıfırlama API'sini yaz	6 saat	Beklemede
Unit testleri oluştur	4 saat	Yapılacak

Scrum Dokümanları - Increment (Artırım / Çalışan Ürün Parçası)

- Increment, bir sprint sonunda ortaya çıkan, **çalışan, test edilmiş, kullanılabilir ürün parçasıdır.** Her sprint sonunda yeni bir "increment" üretilir ve bu, önceki sürüme eklenir.
- **Amaç:**
  - Her sprint sonunda kullanıcıya **gerçek değer sunmak.**
  - Her increment, "kullanıma hazır" olmalıdır (Deploy edilebilir durumda).
- **Özellikleri:**
  - "Done" tanımına (Definition of Done) uygun olmalıdır.
  - Önceki increment'lerle sorunsuz bütünleşmelidir.
  - Sprint Review toplantısında Product Owner ve paydaşlara gösterilir.
- **Örnek:**
  - **Proje:** "Online Ders Platformu"
  - **Sprint 1 Increment:** Giriş ekranı + kullanıcı kaydı
  - **Sprint 2 Increment:** Ders oluşturma + listeleme modülü
  - **Sprint 3 Increment:** Ödev yükleme + not görüntüleme
- Her sprint sonunda ürün **biraz daha tamamlanmış, çalışan bir hale gelir.**

Ek Doküman: Tamamlanma Tanımı (Definition of Done)

- Scrum rehberine göre bu, “artifacts” dışında ama onlarla bağlantılı **ek bir belge**dir.
- “Bir işin ‘tamamlandı’ sayılması için gerekli tüm kriterlerin yazılı tanımıdır.”
- **Örnek:**
  - Kod yazıldı
  - Testleri başarıyla geçti
  - Kod gözden geçirildi (code review)
  - Dokümantasyon güncellendi
  - Dağıtıma hazır hale getirildi
  - Bu listeye uymayan hiçbir iş “tamamlandı” sayılmaz.

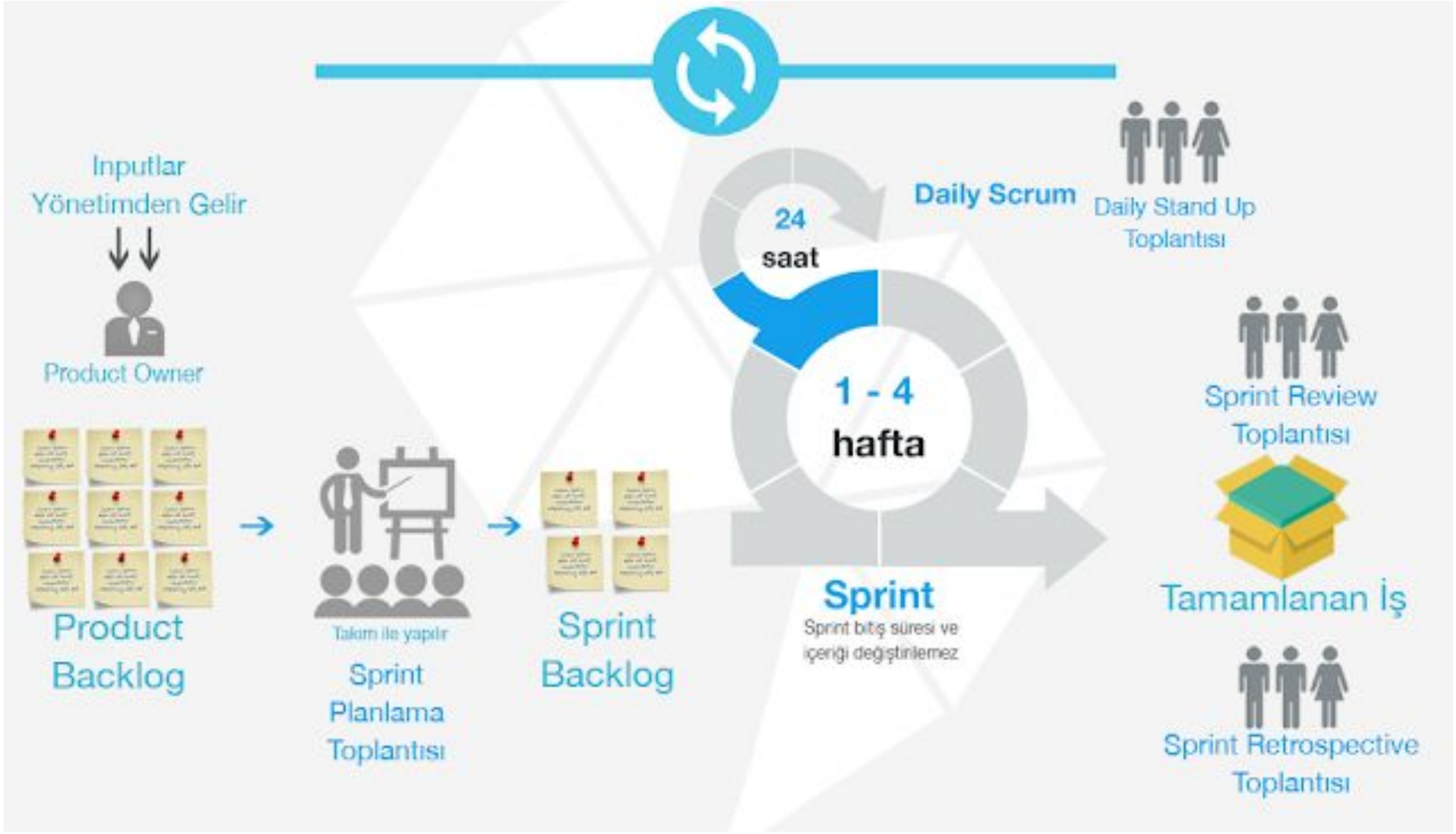
Daily Stand-up (Daily Scrum)

- **Daily Stand-up** veya **Daily Scrum**, Scrum ekibinin her gün **kısa, odaklı ve ayakta yaptığı** toplantıdır. Toplantının amacı, takımın o günkü çalışmasını koordine etmek, engelleri fark etmek ve sprint hedefiyle uyumu sağlamaktır.
- Bu toplantı, **Scrum'ın ritmini** oluşturur. Süre kısadır, genellikle **15 dakika** ile sınırlıdır.
- Amaç detaylı rapor değil, **günlük uyum** sağlamaktır.
- **Amacı**
  - Takımın günlük ilerlemesini gözden geçirmek
  - Engelleri tespit edip erken müdahale etmek
  - Ekip içi iletişimi ve farkındalığı artırmak
  - Sprint hedefiyle senkron kalmak
  - Mikro yönetim değil, *öz-organizasyonu* desteklemek
- **Katılımcılar**
  - **Geliştirme Ekibi:** Ana katılımcıdır.
  - **Scrum Master:** Toplantının formatına uygun ilerlemesini sağlar.
  - **Product Owner:** Gözlemci olarak katılabilir, ama müdahale etmez.
- Daily Scrum bir "raporlama" toplantısı değildir. Scrum Master veya yöneticilere bilgi sunulmaz; **takım kendi içinde konuşur.**

Daily Stand-up (Daily Scrum)

- Her takım üyesi sırayla üç temel soruya cevap verir:
  - **Dün ne yaptım?** → Sprint hedefi doğrultusunda ne tamamladım?
  - **Bugün ne yapacağım?** → Hangi görev üzerinde çalışacağım?
  - **Engelim var mı?** → Hangi sorun ilerlememi engelliyor?
- Bu üç soru takımın **şeffaflığını ve koordinasyonunu** sağlar.
- **Örnek Daily Stand-up Diyaloğu**
  - Ahmet:
    - "Dün kullanıcı girişi modülünü tamamladım. Bugün şifre sıfırlama ekranını tasarlayacağım. Ancak API dokümantasyonuna erişemiyorum, engelim bu."
  - Zeynep:
    - "Dün test senaryolarını yazdım. Bugün veritabanı bağlantılarını kontrol edeceğim. Engelim yok."
  - Ali:
    - "Dün hata kayıtlarını inceledim. Bugün API testlerini çalıştıracağım. Build sistemi yavaş, bu biraz zaman kaybettiriyor."
- Scrum Master not alır, engellerin çözümü toplantı sonrası kısa şekilde planlanır.

Scrum Süreci

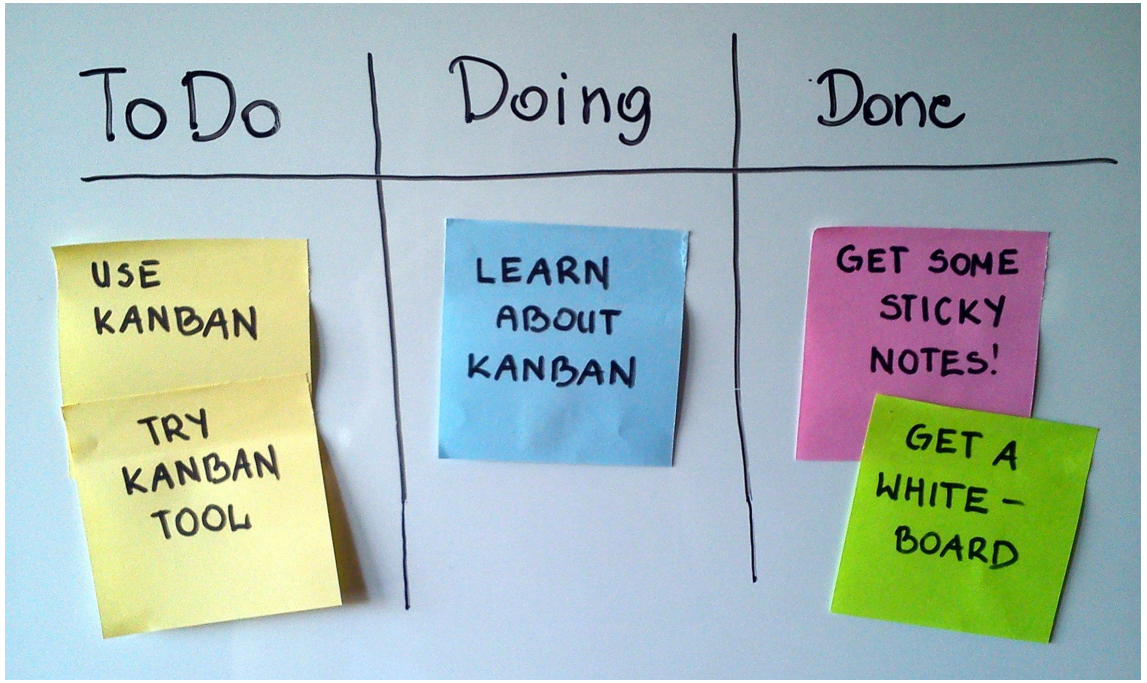


Kanban Yöntemi (Agile – Sürekli Akış Yaklaşımı)

- Kanban, Japonca bir kelimedir ve **“görsel kart”** veya **“işaret levhası”** anlamına gelir.
- İlk kez Toyota’da üretim hattındaki iş akışını yönetmek için kullanılmıştır.
- Yazılım mühendisliğinde ise işlerin görsel olarak izlenmesini, darboğazların fark edilmesini ve sürekli teslimatı hedefler.
- Kanban, **Scrum gibi zaman kutuları (sprint)** kullanmaz.Bunun yerine işler sürekli akar; bir görev tamamlandığında yeni bir görev başlatılır.
- Kanban yaklaşımı dört temel ilkeye dayanır:
  - **Süreci görselleştir (Visualize Workflow):**
    - Tüm işler bir panoda görsel olarak gösterilir.
    - Her iş bir kartla temsil edilir.
  - **Devam eden iş miktarını sınırla (Limit Work in Progress – WIP):**
    - Aynı anda çok fazla iş yapılmaz.
    - WIP limiti, üretkenliği ve kaliteyi artırır.
  - **Akışı ölç ve iyileştir (Manage Flow):**
    - İşlerin “Yapılacak → Yapılıyor → Tamamlandı” akışında tıkanma olup olmadığı sürekli gözlemlenir.
  - **Sürekli geliştirme kültürü oluştur (Continuous Improvement):**
    - Ekip süreçlerini düzenli olarak gözden geçirir ve küçük iyileştirmeler yapar

Kanban Panosu (Kanban Board)

- Kanban yönteminde tüm işler, kartlar hâlinde bir panoda temsil edilir. Bu pano genellikle üç ana sütundan oluşur:
  - Yapılacak (To Do)
  - Yapılıyor (In Progress)
  - Tamamlandı (Done)
- Her kart, bir işi veya görevi temsil eder. Kartlar, süreç boyunca **soldan sağa** taşınır.
- **Kart (Kanban Card) Örneği**
  - Bir kart genellikle şu bilgileri içerir:
  - Görev adı: “Kullanıcı Giriş Ekranı”
  - Sorumlu kişi: “Zeynep”
  - Tahmini süre: “4 saat”
  - Durum: “Yapılıyor”
  - Not: “API entegrasyonu bekleniyor”
- Bu kart, ekip içinde şeffaflığı artırır: herkes kimin ne yaptığını bilir.



WIP Limiti (Work in Progress Limit)

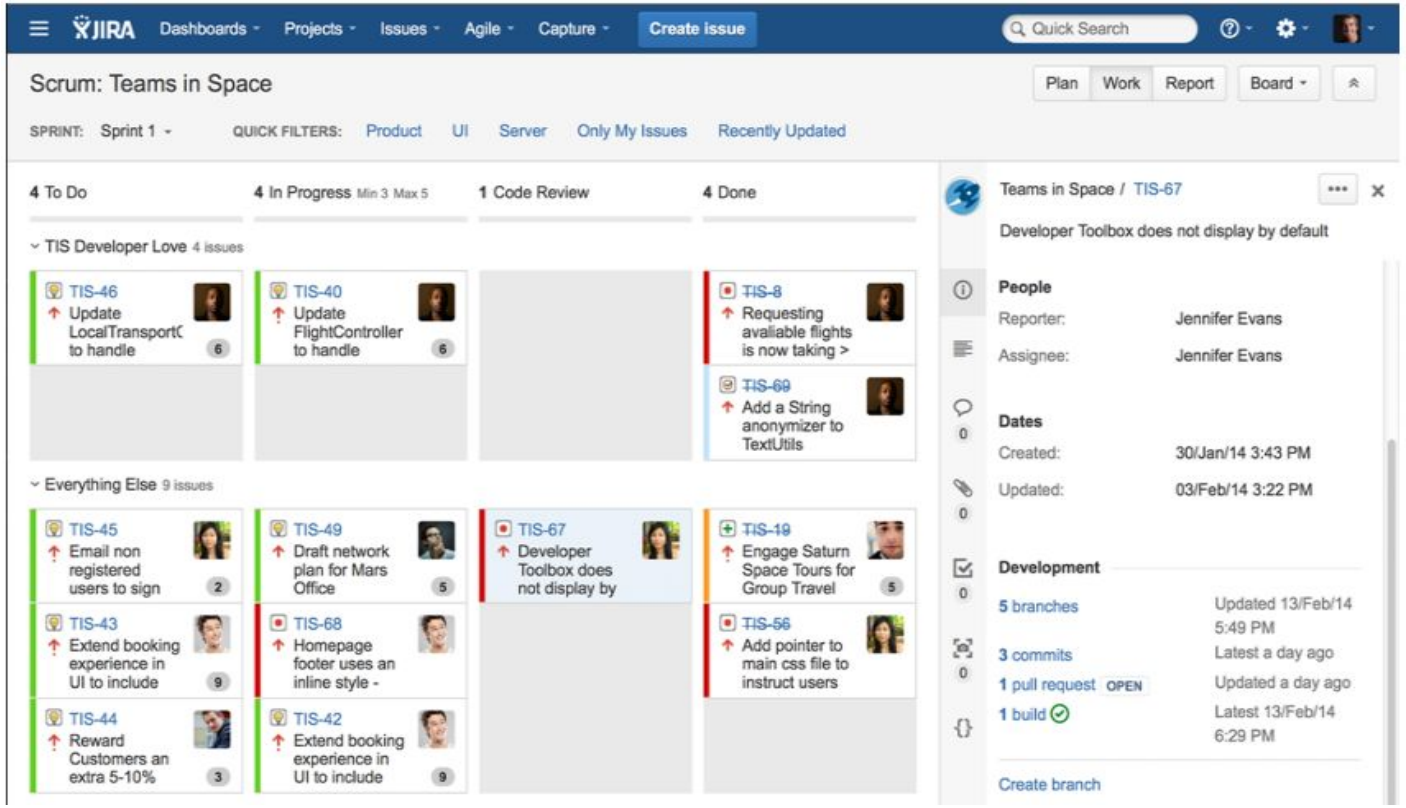
- Aynı anda yapılabilecek iş sayısına getirilen sınırdır.
- Örneğin, “Yapılıyor” sütunu için WIP = 3 denirse, o sütunda en fazla 3 görev bulunabilir.
- Yeni bir işe başlanmadan önce mevcut bir iş tamamlanmalıdır.
- **Neden önemli?**
  - Çok sayıda işi aynı anda yapmak kaliteyi düşürür.
  - WIP limiti ekip üyelerini **odaklanmaya** zorlar.

🧩 To Do	⚙️ In Progress (WIP = 3)	✅ Done
Kullanıcı kaydı ekranı	Giriş API entegrasyonu	Veritabanı bağlantısı
Şifre sıfırlama modülü	Test senaryosu yazımı	Ana sayfa tasarımı
Bildirim sistemi	UI hata düzeltmeleri	-

“Yapılıyor” sütunu dolu. Yeni işe başlanamıyor → **darboğaz var.**
Scrum Master veya takım: Engeli tespit eder ve çözüm önerir (ör. test sürecine destek eklemek).

Jira

- JIRA bir takımın kaynaklarını ve bilgi varlıklarını aynı platformda bir araya getirmesini sağlar.
- Kompleks aktiviteler için farklı departmanların bir araya gelmesi herkesin JIRA'yı kullandığı durumlarda çok daha kolay hale gelir.
- JIRA'daki Agile proje yönetimi modülleri sorumlu ve yöneticilere proje sürecindeki engelleri ortadan kaldırmak ve takımın üretkenliğini arttırmak ve gelişim noktalarına odaklanmak için birinci sınıf bir ortam sağlar.



Performans Ölçümü ve Görselleştirme Araçları

- **Cumulative Flow Diagram (CFD):** İş akışındaki kart sayısının zaman içindeki değişimini gösterir.
- **Lead Time / Cycle Time:** Bir görevin başlangıçtan bitişe kadar geçen süresi.
- **Araçlar:** Trello, Jira, Notion, Asana, ClickUp

Kanban’ın Avantajları ve Sınırlamaları

- Kanban’ın Avantajları ve Sınırlamaları

Avantajlar	Sınırlamalar
Görsel yönetim kolaylığı	Süre tahminlemesi zor olabilir
Esnek ve dinamik yapı	Fazla özgürlük disiplinsizlik yaratabilir
WIP limiti verimliliği artırır	Yeni ekiplerde alışma süreci gerekebilir
Sürekli teslimat	Büyük projelerde Scrum kadar yönlendirici değildir

Scrum ve Kanban Arasındaki Farklar

- Scrum ve Kanban Arasındaki Farklar

Özellik	Scrum	Kanban
Zaman kutuları	Sprint'ler (2–4 hafta)	Sürekli akış
Roller	Product Owner, Scrum Master, Geliştirici	Resmî rol yok
Planlama	Sprint başında yapılır	Her yeni iş için yapılır
WIP Limiti	Belirtilmeyebilir	Temel unsurdur
Değişiklik	Sprint ortasında sınırlı	Her zaman mümkün
Teslimat	Her sprint sonunda	Sürekli
Esneklik	Orta	Yüksek

Agile ile Waterfall Karşılaştırması

- Agile ile Waterfall Karşılaştırması

Özellik	Waterfall	Agile
Süreç yapısı	Lineer	Döngüsel
Geri bildirim	Son aşamada	Sürekli
Esneklik	Düşük	Yüksek
Teslimat	Proje sonunda	Her sprint sonunda
Müşteri rolü	Pasif	Aktif
Belgeleme	Yoğun	Hafif

Agile'ın Zorlukları ve Başarı Faktörleri

- **Zorluklar:**
  - Kurum kültürünün çevik düşünceye uygun olmaması
  - Takım üyeleri arasında iletişim kopukluğu
  - Sürekli değişen öncelikler
- **Başarı Faktörleri:**
  - Güven ortamı
  - Net iletişim
  - Yetkin, motive ekipler
  - Küçük ama ölçülebilir hedefler

- Agile, bir süreç modelinden çok bir **yaklaşım biçimidir.**
- Esnekliğe, sürekli öğrenmeye ve ekip uyumuna dayalıdır.
- Çevik yöntemleri doğru uygulayan kurumlar, değişime en hızlı uyum sağlayan kurumlar haline gelir.

İSTANBUL MEDENİYET ÜNİVERSİTESİ

KATILIMINIZ İÇİN TEŞEKKÜR EDERİM.

YOKLAMA İÇİN İMZANIZI ATMAYI UNUTMAYINIZ.
CLASSROOM ÜZERİNDEN SINIFA DAHİL OLMAYI UNUTMAYINIZ

Sınıf Kodu : pua2tnoe

Dr. Öğr. Üyesi Ertürk ERDAĞI
erturk.erdagi@medeniyet.edu.tr