Yazılım Proje Yönetimi

Proje Yönetim Araçları ve Yaşam Döngüleri

03.03.2025

Halil Bahadır Arın
Kocaeli Üniversitesi
Yazılım Mühendisliği
Kocaeli, Türkiye
bhdrarin@gmail.com

Şamil Alpay
Kocaeli Üniversitesi
Yazılım Mühendisliği
Kocaeli, Türkiye
samila@gmail.com

Mustafa Emirhan Kartal
Kocaeli Üniversitesi
Yazılım Mühendisliği
Kocaeli, Türkiye
mustafak@gmail.com

Ziyaettin Ayerden
Kocaeli Üniversitesi
Yazılım Mühendisliği
Kocaeli, Türkiye
ziyaa@gmail.com

Bu dokuman proje yönetim araçlarını ve proje yönetiminde kullanılan yazılım yaşam döngülerini inceleyerek projemizde kullanılacak yönetim aracının ve yaşam döngüsünün belirlenmesini sağlayacaktır.

I. Yazılım Yaşam Döngüleri

Yazılım yaşam döngüsü (Software Development Life Cycle - SDLC), bir yazılımın fikir aşamasından başlayarak geliştirilmesi, test edilmesi, dağıtılması, kullanılması ve bakımının yapılması sürecini kapsayan aşamalardan oluşan çerçevelerdir

1. Geleneksel SDLC Modelleri

Geleneksel SDLC Modelleri, yazılım geliştirme sürecinde aşamaların sıralı ve düzenli bir şekilde takip edildiği planlı ve yapılandırılmış bir yaklaşımdır. Bu modelde, her aşama tamamlandıktan sonra bir sonraki aşamaya geçilir ve geriye dönüş genellikle zor veya maliyetlidir. Geleneksel SDLC yöntemleri genellikle büyük ölçekli projelerde ve kurumsal yazılım geliştirmede kullanılır.

1.1 Şelale Modeli (Waterfall Model)

Şelale Modeli, yazılım geliştirme sürecinde katı ve sıralı bir yaklaşımı benimseyen bir modeldir. Bu modelde, her aşama tamamlandıktan sonra bir sonraki aşamaya geçilir ve geriye dönüş genellikle mümkün değildir ve maliyetlidir.

Şelale modelinin aşamaları:
Planlama → Analiz → Tasarım → Geliştirme → Test → Dağıtım → Bakım

Şelale modelinin özellikleri:
Aşamalar sıralıdır, geri dönüş zordur.
Dokümantasyon yoğundur.
Küçük veya iyi tanımlanmış projelerde işe yarar.
Esnek değildir, değişiklik yapılması maliyetlidir.

1.2 V Modeli (Verification & Validation Model)

V Modeli (Verification and Validation Model)**,** Şelale Modeli'nin test odaklı bir versiyonudur**.** Bu modelde, her geliştirme aşamasına karşılık gelen bir test aşaması bulunur. Test süreçleri geliştirme aşamalarıyla paralel ilerler, bu yüzden yazılım hataları erken tespit edilir ve düzeltilmesi daha az maliyetli olur.

Aşamalar:

Sol taraf (Doğrulama): Gereksinim analizi → Sistem tasarımı → Mimari tasarım → Modül tasarımı
Sağ taraf (Geçerleme): Birim test → Entegrasyon test → Sistem test → Kabul test

Özellikler:

Her geliştirme aşaması için bir test aşaması bulunur.
Kaliteyi artırır ancak esnek değildir.

1.3 Spiral Modeli

Spiral Modeli**,** şelale modeli ve prototip geliştirme yaklaşımını birleştiren, risk yönetimine odaklanan bir yazılım geliştirme modelidir**.** Bu model, yazılımın geliştirilmesini tekrarlayan döngüler (iterasyonlar) halinde gerçekleştirir ve her döngüde proje daha olgun bir hale gelir.

Aşamalar:

Planlama → Risk Analizi → Mühendislik → Değerlendirme

Özellikler:

Risk yönetimine odaklanır.
Büyük ve karmaşık projelerde kullanışlıdır.
Tekrarlayan döngülerle geliştirme yapılır.
Maliyetli olabilir.

2 Çevik (Agile) SDLC Modelleri

Çevik (Agile) SDLC (Software Development Life Cycle – Yazılım Geliştirme Yaşam Döngüsü) modelleri, geleneksel

yazılım geliştirme süreçlerine göre daha esnek, hızlı ve müşteri odaklı olan yaklaşımlardır.
Geleneksel modellerle arasındaki temel farkı geleneksel SDLC modelleri (Şelale, V Modeli vb.) katı ve sıralı ilerlerken, Çevik SDLC modelleri iteratif (tekrarlayan döngülerle) çalışır.
Hedefi yazılımın hızlı, kaliteli ve değişen müşteri ihtiyaçlarına uyumlu şekilde geliştirilmesini sağlamaktır.

2.1 Agile (Çevik) Model

Agile (Çevik) Model**,** yazılım geliştirme süreçlerinde esneklik, hız ve müşteri odaklılık sağlayan bir yazılım geliştirme metodolojisidir. Geleneksel modellerin aksine, proje boyunca gereksinimlerin değişmesine olanak tanır ve yazılımın küçük parçalar (iterasyonlar) halinde geliştirilmesini sağlar.

Aşamalar:

Planlama → Tasarım → Geliştirme → Test → Yayın → Geri Bildirim

Özellikler:

Esnektir, müşteri geri bildirimlerine göre değişiklik yapılabilir.
Küçük, sık sürümler (iterasyonlar) ile geliştirme yapılır.
Büyük ekiplerde yönetim zor olabilir.

2.2 Scrum

Scrum, Agile (Çevik) yazılım geliştirme metodolojisinin en yaygın kullanılan çerçevelerinden biridir. İteratif ve artımlı bir yaklaşım benimser ve projeyi küçük, yönetilebilir parçalara (sprint) bölerek geliştirme sürecini hızlandırır.
Amacı, esnek ve hızlı bir geliştirme süreci sağlayarak müşteri memnuniyetini artırmak ve ekip verimliliğini en üst düzeye çıkarmaktır.

Bileşenler:

Product Owner (Ürün Sahibi): Gereksinimleri belirler.
Scrum Master: Sürecin doğru işlemesini sağlar.
Development Team: Yazılımı geliştirir.

Aşamalar:

Sprint Planning: Hedefler belirlenir.
Daily Scrum: Günlük 15 dakikalık toplantılar.
Sprint Review: Çıktılar gözden geçirilir.
Sprint Retrospective: İyileştirme analiz edilir.

2.3 Kanban

Kanban, iş süreçlerini görselleştirmek, iş akışını optimize etmek ve ekiplerin verimliliğini artırmak için kullanılan çevik (Agile) bir yönetim metodolojisidir.

Japonca'da "görsel kart" anlamına gelen Kanban, ilk olarak Toyota'nın üretim süreçlerini geliştirmek için kullanılmıştır. Daha sonra yazılım geliştirme ve proje yönetimi gibi farklı alanlara uyarlanmıştır.

Özellikler:

Görselleştirme ve iş akışı optimizasyonuna dayanır.
WIP (Work In Progress) sınırları ile iş yükü kontrol edilir.
Sürekli teslimata uygundur.
Kod basitliği ve sık geri bildirim

2.4 Extreme Programming (XP)

Extreme Programming (XP), müşteri memnuniyetini ve yazılım kalitesini artırmaya odaklanan bir Çevik (Agile) yazılım geliştirme metodolojisidir. XP, hızlı değişen gereksinimlere uyum sağlamak, sürekli geri bildirim almak ve yazılım geliştirme sürecini iyileştirmek için tasarlanmıştır.
XP'nin en büyük farkı, aşırı derecede tekrar eden geliştirme döngüleriyle (iterasyonlarla) çalışması ve en iyi yazılım mühendisliği uygulamalarını uç noktaya kadar kullanmasıdır.

Özellikler:

Çift programlama (Pair Programming)
Test güdümlü geliştirme (TDD)
Sürekli entegrasyon
Kod basitliği ve sık geri bildirim

2.5 Lean Yazılım Geliştirme

Lean Yazılım Geliştirme (Yalın Yazılım Geliştirme), hızlı, verimli ve müşteri odaklı yazılımlar geliştirmek için kullanılan bir metodolojidir. Toyota'nın üretim sürecinden (Lean Manufacturing) esinlenerek yazılım dünyasına uyarlanmıştır.
Lean, israfı (waste) minimize etmeyi, değer yaratmayı ve sürekli iyileştirmeyi hedefler. Agile (Çevik) metodolojisinin bir alt disiplini olarak da görülebilir.

3. Hibrit Modeller

Hibrit (Hybrid) Yazılım Geliştirme Modelleri, farklı SDLC (Yazılım Geliştirme Yaşam Döngüsü) yaklaşımlarının en iyi yönlerini birleştirerek oluşturulan

modellerdir. Geleneksel ve Çevik (Agile) metodolojilerin avantajlarını bir araya getirerek esneklik, kalite ve hızın dengelenmesini sağlarlar.

Hibrit modeller, hem planlama ve dokümantasyonun önemli olduğu büyük ölçekli projeler için hem de hızlı teslimat gerektiren dinamik projeler için uygundur.

3.1 V-Model + Agile (Hibrit V-Model)

V-Model + Agile, yazılım geliştirme süreçlerinde V-Model'in disiplinli doğrulama ve geçerleme süreçlerini Agile metodolojisinin esnekliği ile birleştiren bir hibrit modeldir. Bu yaklaşım, yazılım geliştirme sürecinde hem sıkı test süreçlerini hem de iteratif ve müşteri odaklı geliştirmeyi aynı anda uygular. Özellikle sağlık, finans ve savunma gibi regülasyonlara tabi sektörlerde tercih edilir.

Bileşenler:

Product Owner (Ürün Sahibi): Gereksinimleri belirler ve müşteri beklentilerini yönetir.
Scrum Master: Agile süreçlerin verimli işlemesini sağlar.
Development Team: Kod geliştirme, test ve entegrasyon işlemlerini gerçekleştirir.
Test Engineers: V-Model'in test aşamalarını uygular.

Aşamalar:

Gereksinim Analizi: Kullanıcı hikayeleri oluşturulur, Product Backlog hazırlanır.
Sistem Tasarımı: Sprint planlaması yapılır, geliştirme süreci belirlenir.
Mimari ve Modül Tasarımı: Yazılımın teknik altyapısı oluşturulur.
Kodlama & Birim Testleri: Agile sprintler içinde geliştirme yapılırken, birim testleri uygulanır.
Bütünleşme (Entegrasyon) Testleri: Kod parçaları sürekli entegrasyon (CI) ile test edilir.
Sistem Testleri: Her iterasyon sonunda yazılımın tamamı kapsamlı test edilir.
Kabul Testleri: Kullanıcı geri bildirimleri değerlendirilerek yazılım iyileştirilir.

3.2 Agile + Waterfall (Hibrit Agile-Waterfall)

Agile + Waterfall, yazılım geliştirme süreçlerinde geleneksel Şelale (Waterfall) modelinin planlı yapısını Agile metodolojisinin esnekliği ile birleştiren hibrit bir yaklaşımdır. Büyük ölçekli projelerde, özellikle regülasyon gerektiren veya uzun vadeli planlama gerektiren durumlarda tercih edilir. Agile'ın iteratif yapısı sayesinde değişikliklere hızlı uyum sağlanırken, Waterfall modelinin belirli aşamalara sahip planlı süreci korunur.
Bileşenler:

Project Manager: Genel proje yönetimini ve Waterfall aşamalarını denetler.
Product Owner (Ürün Sahibi): Gereksinimleri belirler ve Agile süreçlerini yönetir.
Scrum Master: Agile metodolojisinin süreçlerini yönetir.
Development Team: Yazılım geliştirme ve test süreçlerini yürütür.

Aşamalar:

Planlama (Waterfall): Proje gereksinimleri detaylı olarak belirlenir.
Tasarım (Waterfall): Yazılımın genel mimarisi oluşturulur.
Geliştirme (Agile): Yazılım geliştirme süreci iteratif sprintlerle ilerler.
Test (Agile & Waterfall): Agile'da her iterasyon sonunda test yapılırken, final aşamada Waterfall'a uygun geniş kapsamlı testler uygulanır.
Kabul & Dağıtım (Waterfall): Ürün, müşteri ve kalite standartlarına göre nihai testlerden geçirilerek teslim edilir.
Bu model, kapsamlı planlama gerektiren ancak değişken gereksinimlere hızlı uyum sağlamak isteyen projeler için uygundur.

3.3 Agile + Spiral (Hibrit Agile-Spiral)

Agile + Spiral, risk yönetimi odaklı Spiral modelini, Agile'ın iteratif geliştirme yapısıyla birleştiren bir hibrit modeldir. Bu model, özellikle yüksek risk taşıyan projelerde, müşteri geri bildirimlerinin ve sık test süreçlerinin önemli olduğu durumlarda kullanılır. Spiral modelin risk analizi aşaması, Agile metodolojisinde her sprintte uygulanarak sürekli iyileştirme sağlanır.

Bileşenler:

Product Owner: Müşteri gereksinimlerini yönetir.
Scrum Master: Agile süreçlerin düzenli ilerlemesini sağlar.
Risk Analyst: Spiral modelin risk analizlerini yürütür.
Development Team: Yazılım geliştirme ve test süreçlerini gerçekleştirir.

Aşamalar:

Planlama & Risk Analizi (Spiral): Proje başlangıcında risk değerlendirmesi yapılır.
Geliştirme & Test (Agile): Yazılım, sprintler halinde geliştirilir ve her sprint sonunda test edilir.
Değerlendirme (Spiral): Her sprint sonunda, riskler ve müşteri geri bildirimleri gözden geçirilerek gerekli iyileştirmeler yapılır.

Yineleme & Teslimat: Yeni sprintlerde riskler minimize edilerek geliştirme devam eder.
Bu model, özellikle büyük ve riskli projelerde kullanılarak hem güvenilir hem de esnek bir geliştirme süreci sunar.

II. Proje Yönetim Araçları

Proje yönetim araçları, ekiplerin projelerini planlamasına, yürütmesine ve izlemesine yardımcı olan yazılımlardır. Bu araçlar, görev yönetimi, zaman takibi, iş birliği, kaynak planlama ve raporlama gibi özellikler sunarak projelerin daha verimli ve organize bir şekilde ilerlemesini sağlar.

1 Jira

Atlassian tarafından geliştirilmiş, yazılım geliştirme ve Agile metodolojisine odaklı bir proje yönetim aracı. Genellikle büyük yazılım ekipleri, Scrum ve Kanban gibi Agile yöntemleri kullanan firmalar tarafından tercih edilir.

Özellikler:

Scrum ve Kanban desteği: Sprint planlaması, backlog yönetimi.

İleri düzey hata takibi: Yazılım projeleri için bug ve issue tracking.

Gelişmiş raporlama ve analiz: Burndown chart, velocity chart gibi metrikler.

Jira Query Language (JQL): Güçlü filtreleme ve arama özellikleri.

Üçüncü taraf entegrasyonlar: Confluence, Bitbucket, GitHub vb.

Artıları:

Agile metodolojileri destekler.

Güçlü özelleştirme seçenekleri.

Büyük ekipler için uygundur.

Zengin entegrasyon ekosistemi.

Eksileri:

Kullanımı karmaşık olabilir, öğrenme eğrisi yüksektir.

Küçük ekipler için gereksiz derecede detaylı olabilir.

Ücretli sürüm maliyetlidir.

2 Azure Boards

Microsoft'un sunduğu, özellikle Azure DevOps ile entegre çalışan bir proje yönetim aracı. Genellikle yazılım geliştirme ekipleri ve Microsoft teknolojilerini kullanan firmalar tarafından tercih edilir.

Özellikler:

Kanban ve Scrum desteği: Sprint planlaması, görev yönetimi.

Azure DevOps entegrasyonu: Git, CI/CD pipeline ile entegre çalışma.

Gelişmiş raporlama ve analitik: Work item tracking, burndown chart vb.

Özelleştirilebilir iş akışları: Şirket süreçlerine uygun hale getirilebilir.

Güçlü güvenlik ve erişim kontrolleri.

Artıları:

Azure ve DevOps araçları ile güçlü entegrasyon.

Microsoft ekosistemi ile tam uyum.

Büyük ekipler için uygun.

Güçlü güvenlik ve erişim yönetimi.

Eksileri:

Microsoft ekosistemine bağımlılık gerektirir.

Kullanımı yeni başlayanlar için karmaşık olabilir.

Küçük ekipler için fazla detaylı olabilir.

3 Trello

Kullanımı kolay, görselliği ön planda olan, kart tabanlı bir proje yönetim aracı.
Özellikle küçük ekipler, bireysel kullanıcılar ve basit iş akışları için idealdir.

Özellikler:

Kart ve pano bazlı organizasyon: Görevleri sürükle-bırak yöntemiyle yönetme.

Check-list ve etiketleme: Görevlerin detaylandırılması.

Basit entegrasyonlar: Slack, Google Drive, Dropbox gibi araçlarla çalışma.

Otomasyon desteği: Butler ile otomatik işlemler.

Artıları:

Kullanımı çok kolay ve hızlı öğrenilir.

Görsel ve esnek bir arayüz.

Küçük projeler için mükemmel bir çözüm.

Ücretsiz sürümü oldukça yeterli.

Eksileri:

Büyük ve karmaşık projeler için yetersiz kalabilir.

İleri seviye raporlama ve analiz özellikleri sınırlıdır.

İş akışlarının özelleştirilmesi sınırlıdır.

4 Asana

Takım bazlı görev yönetimine odaklanan, sezgisel bir proje yönetim aracıdır. Orta ve büyük ölçekli ekiplerin iş takibi için kullanılır.

Özellikler:

Görev ve proje yönetimi: Liste, pano, takvim görünümü.

İş akışı otomasyonu: Tekrarlayan görevleri otomatikleştirme.

Hedef ve kilometre taşı belirleme.

Detaylı bildirim ve hatırlatmalar.

Slack, Google Drive ve diğer entegrasyonlar.

Artıları:

Kullanıcı dostu arayüz ve kolay kullanım.

Esnek görev yönetimi ve görünüm seçenekleri.

İş akışı otomasyonu ile verimlilik artırma.

Ücretsiz versiyonu küçük ekipler için yeterli.

Eksileri:

Gelişmiş raporlama özellikleri sınırlıdır.

Jira ve Azure Boards gibi yazılım geliştirme odaklı değildir.

Büyük projelerde bazen karışık olabilir.

5 Monday.com

Görselliğe önem veren, geniş kullanım alanına sahip bir proje yönetim platformu. Çeşitli sektörlerdeki ekiplerin iş takibini kolaylaştırır.

Özellikler:

Esnek ve özelleştirilebilir arayüz: Farklı iş akışlarına uyarlanabilir.

Kanban, Gantt, tablo ve takvim görünümü.

Otomasyon ve entegrasyon desteği: Zapier, Slack, Google Drive ile uyumlu.

Detaylı kullanıcı izinleri ve güvenlik yönetimi.

Artıları:

Kullanıcı dostu ve görsel arayüz.

Çeşitli sektörlere uygun.

Geniş entegrasyon seçenekleri.

Esnek iş akışı ve özelleştirme.

Eksileri:

Ücretsiz sürümü çok sınırlıdır.

Büyük ekipler için maliyetli olabilir.

Çok fazla özelleştirme bazen karmaşıklık yaratabilir.

III. Sonuç

Öğrenci takip programını geliştirirken, değişken kullanıcı ihtiyaçlarına hızlı adapte olabilen, esnek ve verimli bir geliştirme süreci yürütmek istiyoruz. Projemizde, bireyselleştirilmiş çalışma planları, deneme sınavı analizleri ve motivasyon sistemleri gibi sürekli güncellenmesi gereken bileşenler yer almaktadır. Bu nedenle, değişiklikleri kolayca yönetebilmek ve her aşamada çalışan bir sistem sunabilmek için Agile metodolojisinin en popüler çerçevelerinden biri olan Scrum modelini tercih ettik.

Scrum, projeyi küçük parçalara (sprintler) bölerek geliştirmeyi sağlar. Böylece her sprint sonunda belirli bir özellik tamamlanır, test edilir ve kullanıcı geri bildirimleri doğrultusunda geliştirilir. Örneğin, ilk sprintte kullanıcı kayıt sistemi ve öğrenci profili geliştirilirken, sonraki sprintlerde çalışma programı oluşturma, deneme analizleri ve ödüllendirme mekanizmaları gibi özellikler eklenecektir. Bu sayede, projenin tamamlanmasını beklemeden erken aşamalarda işlevsel bir sistem sunabilir ve kullanıcı geri bildirimlerine göre iyileştirme yapabiliriz.

Görev dağılımının net olması ve ekibin organize bir şekilde ilerleyebilmesi için Scrum'un kısa toplantılar (Daily Scrum) ile ekip üyelerinin birbirlerinin ilerlemesini takip etmesini sağlaması, ekip içi iletişimi güçlendirir ve işlerin aksamasını önler. Ayrıca, her sprint başında yapılacak işler belirlendiği için hangi özelliğin hangi aşamada olduğu kolayca takip edilebilir.

Scrum modeliyle birlikte, proje yönetimini kolaylaştırmak için Trello gibi araçlar kullanarak görevleri daha düzenli şekilde yönetmeyi planlıyoruz.

Alternatif modelleri değerlendirdiğimizde, Waterfall (Şelale) modeli değişiklik yapmayı zorlaştırdığı için uygun değildir. Spiral modeli, büyük ölçekli ve uzun vadeli projeler için daha uygundur. V-Model, her aşamada test süreçlerini gerektirdiği için süreci uzatır ve değişiklikleri yönetmeyi zorlaştırır. Kanban, görev önceliklerini belirlemede esneklik sunmasına rağmen, sprint planlamasının olmaması ekip için takibi zorlaştırabilir.

Bu nedenlerle, Scrum modeli, hem proje gereksinimlerimize hem de ekibimizin çalışma yapısına en uygun seçimdir. Esnekliği, hızlı geliştirme döngüsü ve ekip içi iletişimi güçlendirmesi sayesinde projemizi başarılı bir şekilde tamamlamamıza yardımcı olacaktır.

Projemizi geliştirirken takım içi koordinasyonu sağlamak, görevleri düzenli bir şekilde takip etmek ve ilerlemeyi şeffaf hale getirmek için kullanımı kolay, esnek ve görsel bir yönetim aracı olan Trello'yu tercih ettik.

Trello, "Kanban" sistemine dayalı bir görev yönetim aracıdır ve kartlar, listeler ve panolar kullanarak iş akışlarını düzenlemeye olanak tanır. Scrum modeliyle uyumlu olması, sprint süreçlerimizi verimli bir şekilde yönetmemize yardımcı olacaktır. Örneğin, her sprint için ayrı bir liste oluşturarak, "Yapılacaklar" (To Do), "Devam Edenler" (In Progress) ve "Tamamlananlar" (Done) gibi bölümler kullanabiliriz. Böylece ekip üyeleri, hangi görevlerin tamamlandığını ve hangi işlerin devam ettiğini kolayca görebilir.

Trello'yu tercih etme sebeplerimizden biri de basit arayüzü ve kullanım kolaylığıdır. ekip olarak, karmaşık proje yönetim araçları yerine, hızlı adapte olabileceğimiz ve ek yük getirmeyen bir platforma ihtiyacımız var. Trello, sürükle-bırak yöntemiyle görevleri taşımaya olanak sağladığı için, sprint sürecinde görevleri kolayca güncelleyebilir ve ilerlemeyi net bir şekilde takip edebiliriz.

Ayrıca, Trello sayesinde görevleri ekibe atayabilir, son teslim tarihleri belirleyebilir ve açıklamalar ekleyebiliriz. Örneğin, "Kullanıcı Kayıt Sistemi" görevini bir ekip üyesine atayarak, ilgili kişi tamamladığında kartı "Tamamlandı" listesine taşıyabilir. Böylece ekip içindeki herkes hangi görevlerin tamamlandığını ve hangi görevlerin üzerinde çalışıldığını görebilir.

Alternatif proje yönetim araçlarını değerlendirdiğimizde, Jira, büyük ve karmaşık projeler için daha uygun olup, junior bir ekip için gereksiz derecede detaylı olabilir. Asana, görev yönetimi açısından güçlü olsa da, Trello kadar basit ve görsel değildir. Notion, dökümantasyon açısından güçlü olsa da, görev takibi için Trello kadar pratik değildir.

Sonuç olarak, Scrum modeline uygunluğu, kullanım kolaylığı ve görsel takip imkanı nedeniyle Trello'yu tercih ettik. Trello sayesinde sprint süreçlerimizi organize edebilir, ekip içi görev dağılımını netleştirebilir ve proje ilerlemesini kolayca takip edebileceğimize karar verdik.