

**YAZILIM GELİŞTİRME VE YAŞAM DÖNGÜLERİ**

**HAZIRLAYAN**

**BEYDA KILIÇOĞLU – 210601025**

**HESAP LİNKLERİ:**

**Linkedİn:** [**https://www.linkedin.com/in/beyda-k%C4%B1l%C4%B1%C3%A7o%C4%9Flu-777610224/**](https://www.linkedin.com/in/beyda-k%C4%B1l%C4%B1%C3%A7o%C4%9Flu-777610224/)

**GitHub:** [**https://github.com/beida-ce**](https://github.com/beida-ce)

**Medium:** [**https://medium.com/@kilicoglubeyda**](https://medium.com/@kilicoglubeyda)

**İÇİNDEKİLER**

**1.GİRİŞ**

**2.Yazılım Yaşam Döngüleri**

**2.1. Yazılım Yaşam Döngü Modellerinin Adımları**

**2.1.1. Gereksinim Fazı**

**2.1.2. Analiz Fazı**

**2.1.3. Tasarım Fazı**

**2.1.4. Gerçekleştirme Fazı**

**2.1.5. Test-Teslim Fazı**

**2.1.6. Teslim Sonrası Bakım Fazı**

**2.1.7. Emeklilik Fazı**

**3.Yazılım Yaşam Döngü Modellerinin Sınıflandırılması ve Karşılaştırılmaları**

**3.1. Şelale Modeli**

**3.2. V Modeli**

**3.3. Helezonik (spiral) Model**

**3.4. Artımsal Geliştirme Modeli**

**3.5. Kodla-Düzelt Modeli**

**4.Hangi Projede Hangi Model Kullanılmalı?**

**5.Çevik Yazılım Geliştirme (AGİLE)**

**5.1. Çevik yazılım geliştirme prensipleri**

**5.2. Çevik yazılım geliştirme yöntemleri**

**5.2.1. Scrum nedir? Neden popüler?**

**6.KAYNAKÇA**

**1.GİRİŞ**

Bu makalede yazılım yaşam döngüleri ve geliştirme süreçleri hakkında bilgi verilecektir.

**2. Yazılım Yaşam Döngüleri**

Yazılım yaşayan bir üründür. Geliştirilmekte olan yazılım ürününün planlamasından teslimine kadar olan sürece “yazılım yaşam döngüsü” denir. Yazılım yaşam döngüleri; daha kaliteli, tasarlanması, uygulanması, test edilmesi daha kolay yazılımlar geliştirmek için kullanılmaktadır. Yazılımların yaşam döngüleri birkaç temel basamaktan oluşur. Basamaklar, her zaman doğrusal veya tek bir şekilde takip edilip uygulanmaz. Çünkü yazılımın geliştirilmeye başlandığı andan itibaren gereksinimler ve işlevsellikler değişebilir. Bu nedenle basamaklar bir döngü olarak ele alınmalıdır. Basamakları takip etmedeki ve uygulamadaki farklar “Yazılım Yaşam Döngü Modellerini” ortaya çıkarmıştır. Bu modellerdeki ortak basamaklar temel olarak gereksinim, analiz, tasarım, gerçekleştirme, test, teslim, teslim sonrası bakım ve emeklilik olarak adlandırılabilir.

**2.1. Yazılım Yaşam Döngü Modellerinin Adımları**

**2.1.1. Gereksinim Fazı**

Gereksinim fazında müşteri talepleri toplanır, yazılım ürününün özellikleri ayrıntılarıyla belirlenir, ve bir gereksinim listesi oluşturulur. Gereksinim listesi göz önüne alınarak kullanılacak kaynaklar, mali süreç, projenin hedef kitlesi vb. belirlenir. Bu planlamadaki temel amaçlardan birisi de projeyi istenilen süreç içerisinde tamamlamak olduğundan özellikle projenin yapım aşamalarının zamanlamasına dikkat edilmelidir

**2.1.2. Analiz Fazı**

Analiz, ihtiyaçların ve beklentilerin detaylı olarak tanımlamasının yapıldığı basamaktır. Burada ihtiyaç ve beklenti, yazılımın gerçekleştirmesinin istendiği temel görevdir. Bu ihtiyaç ve beklentiler aşağıdaki özellikleri taşımalıdır.

* **Tutarlılık:** Bir ihtiyaç diğer ihtiyaç ve beklentilerle zıt düşmemelidir.
* **Gerçekleştirilebilme:** Eldeki kaynakların projenin gerçekleştiriminde yeterli olması.
* **Çözümlerin değil isteklerin tanımlanıyor olması:** Analizin hedefini belirttiğinden önemlidir.
* **Yeri doğru olmalı:** Bir ihtiyacın gereğinden fazla ön planda tutulması veya gereğinden altta tanımlanması sistem yapısına uygun değildir.

**2.1.3. Tasarım Fazı**

Gereksinim fazında istenen ve analiz fazında belirlenmiş olan gereksinimleri sağlayacak ürünün tasarlanmasıdır. Tasarım fazı iki basamaktan oluşur. Birinci basamak olan “Mimari Tasarım”, sistemin yapısının oluşturulduğu basamaktır. UML grafikleri (Birleşik Modelleme Dili), nesne modelleri ve akış şemaları kullanılabilir. İkinci basamak olan “Detaylı Tasarım” sırasında, veri yapıları (structlar), ekran tasarımları gibi detaylara inilir. Ayrıntılıdır.

**2.1.4. Gerçekleştirme Fazı**

Talepler doğrultusunda yapılan tasarıma uyun geliştirme araçlarıyla tasarımı ürüne dönüştürme işlemi yapılır. Rapor yazılması, kodlama yapılması, kullanıcı ara yüzü oluşturulması, veri tabanlarının oluşturması gibi işlemler tamamlanır. Ürünün ilk çıktıları ortaya çıkar. Modüllere ait birim testleri bu aşamada yapılabilir.

**2.1.5. Test-Teslim Fazı**

Test aşamasında gerçek veri kullanılarak geliştirilen ürün test edilir ve gerekiyorsa düzeltmeler yapılarak ürünün son hali müşteriye teslim edilir.

**2.1.6. Teslim Sonrası Fazı**

Teslim sonrası yapılan herhangi bir değişiklik “teslim sonrası bakım” olarak adlandırılır. Düzeltici bakım ve özelliklerin arttırımı olmak üzere iki aşamadan oluşur.

* **Düzeltici Bakım:** Sistem hatalarını düzeltmek için yapılır.
* **Özelliklerin Arttırımı:** Müşterinin isteğine göre yazılıma uyarlanan içerikler.
  + **Mükemmelleştirici Bakım:** Yazılım ürününün yararlılığını arttırmayı hedefler.
  + **Uyarlanabilir Bakım:** Yazılım ortamı değiştiğindemeydana gelen değişiklik ve sorunlara karşı yapılır.

**2.1.7. Emeklilik Fazı**

Yazılıma olan ihtiyaç bittiğinde veya artık geliştirmek yerine yeni bir ürün ortaya çıkarmak daha mantıklı olduğunda ürün artık emekliye ayrılır.

**3.Yazılım Yaşam Döngü Modellerinin Sınıflandırılması ve Karşılaştırılmaları**

Her bir yazılım yaşam döngü modelinin kendi içinde avantajları ve dezavantajları vardır. Bu avantajlar doğrultusunda hangi modelin hangi projede kullanıma daha uyun olduğu bir sonraki başlıkta anlatılacaktır.

**3.1. Şelale Modeli**

Şelale modeli bir geleneksel yazılım geliştirme modelidir. Diğer geleneksel yazılım geliştirme modellerinde de adımlar şelale modelinde olduğu gibi doğrusal olarak takip edilir. Her adım bir önceki adımın çıktılarını kullanır. Bir sonraki basamağa geçmek için çıktı oluşması gerektiğinden her bir adım eksiksiz takip edilip tamamlanmalıdır. Yani yazılım geliştirme süreci doğrusaldır. Değişen beklenti ve ihtiyaçlar olduğunda tüm adımlar baştan takip edilmeli ve ürün geliştirilmelidir. Yazılım geliştirme model adımlarının bu kadar kesin sınırlarla birbirinden ayrılması gerçekçi değildir. Yazılım projeleri bu modelin aksine çok daha esnek olduğundan adımların bu şekilde takip edilmesi zordur. Projelerde bu sınırlar kaybolabilir. Değişikliğe açık bir model değildir çünkü safhalar arası geri dönüşler yetersizdir. Bu da yapılan hataların düzeltilmesini zorlaştırır. Düşük maliyetlidir. Büyük projelerde kullanıma uygun olmadığı gibi, küçük projelerde maliyetten kısacaktır. Ürünün kullanıma sunulması uzun zaman alabilir. Müşteri sadece planlamanın başında projeye dahildir. Müşterinin beklentilerini ve ihtiyaçlarının değişmesi adımların tekrar tekrar uygulanmasını gerektireceğinden maliyeti arttıracaktır. Proje bu şartlarda geliştirilse bile müşterinin beklentilerini tatmin etmeyebilir.

**3.2. V Modeli**

Adını hem şeklinden, hem de aşamaları olan “Verification and Validation (onaylama ve doğrulama)”dan alır.

* **Verification:** Gereksinim analizi basamağında, sistem gereksinimleri kullanıcının ihtiyaçları incelenerek toplanır. Müşterilerle görüşülür ve gereksinim dokümanı oluşturulur. Sistem tasarımı basamağında yazılımı geliştirecek kişinin veya ekibin gereksinim basamağında oluşturulan dokümanları anlaması ve sistemin temelini, iskeletini planlaması söz konusudur. Mimari tasarım basamağında sistem tasarımı basamağındaki planlama geliştirilir. Ara yüz kullanımları, modüller, veri tabanı tabloları oluşturulur. Modül tasarımı basamağı da mimari tasarımın içinde bir basamak olarak düşünülebilir. Halihazırda tasarlanan sistem daha küçük parçalara ayrılıp planlanmıştır ve artık kodlanmaya başlanabilir.
* **Validation:** Birim test basamağında oluşturulan ve kodlanan modüller kendi içerilerinde test edilir. Hatalar tespit edilip düzeltilir ve modüller bağlanmaya hazır hale getirilir. Entegrasyon test basamağı mimari tasarım basamağı boyunca uygulanır. Modüllerin birbiri ile uyumunda sıkıntı olup olmadığı test edilir. Sistem test etme basamağı, sistem tasarımı basamağı boyunca uygulanır. Bu basamakta geliştirilen ürünün fonksiyonel ve fonksiyonel olmayan işlevleri yerine getirip getirmediğine bakılır. Kullanıcı kabul testi basamağı, gereksinim analizi basamağı boyunca uygulanır. Müşteri

gereksinimlerinin karşılandığından ve kullanıma hazır olduğundan emin olur.

V modelinin avantajları arasında disiplinli olması, gereksinimlerin anlaşıldığı daha küçük çaplı projelerde iyi çalışması, anlaşılır ve kullanımının kolay olması sıralanabilir. Ancak bu model daha karmaşık ve nesne yönelimli projeler için uygun olmadığı gibi, uzun soluklu projelerde de zayıf kalacaktır.

**3.3. Helezonik (spiral) Model**

Bu model risk analizini ön planda tutar. Yinelemeli artımsal bir yaklaşımdır. Görsel olarak bir spiral düşünüldüğünde bu spiral döngülerden oluşacaktır. Her bir döngü kendi içerisinde risk analizini barındırır. Spiral döngü modeli kendi içerisinde 4 adım barındırır. Planlama aşamasında döngü sonunda üretilecek ara ürün için planlama yapılır, amaç belirlenir, ve bir önceki döngüde oluşan ara ürün ile yeni döngünün sonunda oluşacak ara ürün ile bütünleştirilir. Bu modelde ön planda tutulan risk analizinin yapıldığı “risk analizi” aşamasında risk seçenekleri araştırılır ve belirlenir. Üretim aşamasına gelindiğinde, planlama aşamasında belirlenen ara ürün oluşturulur. Her bir döngünün son aşaması olan “kullanıcı değerlendirmesi” aşamasında ara ürün müşterinin denemesine sürülür. Anlaşılacağı üzere müşteri, geliştirmenin her aşamasında geliştirici ve ürünle iç içedir. Gelişen ve değişen beklentiler kolaylıkla eklenip çıkarılabileceğinden spiral model tercih edilebilmektedir. Bu modelin dezavantajları arasında karmaşık yönetime sahip olması, küçük veya düşük riskli projeler için gereğinden fazla kaynak ve bütçe kullanımı, çok sayıda ara aşama için çok sayıda dokümantasyon gerekmesi olarak sıralanabilir.

**3.4. Artımsal Geliştirme Modeli**

Bu yazılım yaşam döngü modelinde geliştirme teslimat parçalara bölünür. Her bir teslim beklenti ve ihtiyaçların bir kısmını karşılar. Müşteri gereksinimleri önceliklendirilir ve en önemli gereksinimler ilk sırada teslim edilir. Birbirine bağlı gereksinimler de buna dayanarak belirli bir sırada teslim edilmektedir. Beklenti ve gereksinimlerdeki değişimlerle ilerideki teslimlerde ilgilenilir. Her bir teslim “sürüm” olarak adlandırılabilir. Her bir sürüm daha büyük bir ürünü kapsar ve işlevin arttığı yorumu yapılabilir. Müşteri kullanıma başlayabilir, üretim tüm gereksinimler karşılanana kadar devam edecektir. Uzun soluklu ve büyük projelerde bu model kullanıma uygundur. Müşteriye verilen ilk teslimler, geliştirici ekip için prototip olarak kullanılabilir. Öncelikli gereksinimler üzerinde daha çok durulur, daha çok teste uğrarlar. Müşteri kullanıma başlayabileceğinden sistemin işe yarayıp yaramadığı erkenden anlaşılabilir. Bu şekilde bütçeden kazanç sağlanır. Risk yönetimi kolaylığı vardır. Yüksek riskli kısımların önceden tanımlanıp tamamlanması müşteri, geliştirici ve yazılım ürünü için daha sağlıklıdır. Dezavantajları arasında fazla kaynak gereksinimi, projenin sonunun bilinmemesi ve küçük projeler için uygun olmayışı vardır.

**3.5. Kodla-Düzelt Modeli**

Küçük, yaklaşık birkaç yüz satırlık projelerde kullanılabilir. Dokümantasyon oluşturulmadan direkt ürün geliştirmeye gidilir. Dokümantasyon oluşturulmadığından bakım aşamasının uygulanması çok zordur. İlk versiyon gerçekleştirilir, müşteri gereksinimleri karşılanana kadar ürün düzenlenir. Yaşam döngüsünün en sonunda ürün emekliye ayrılır. Yazılım geliştirmenin en kolay yoludur, ancak en maliyetlisidir. Dokümantasyon olmadan her seferinde bakım yapmak çok zor ve maliyetlidir. Kolay bir yol olduğu için küçük veya tecrübesiz şirketler tarafından tercih edilebilir.

**4.Hangi Projede Hangi Model Kullanılmalı?**

Projeler boyutlarına ve geliştirilmesinin aldığı zamana göre sınıflandırılabilir. Bu 5 sınıfı büyükten küçüğe Grand, Large, Medium, Moderate, Small olarak sıralarız. Grand ve Large gibi büyük ve uzun soluklu projelerde spiral model gibi risk analizinin fazla olduğu modeller kullanılabilir. Fazla bütçe ayırılsa da ileride oluşabilecek daha büyük bütçe açıklarının ihtimallerini en aza indirebilir. Bu yüzden kullanımı mantıklıdır. Artımsal geliştirme modeli de tıpkı spiral modelde olduğu gibi büyük projeler için kullanıma daha uygundur. Riskli gereksinimlerin önceden tamamlanması ve projenin müşteriye erken teslimi sayesinde hem ürün için riski aza indirir, hem de beklentilerin karşılanıp karşılanmadığı erken müşteri tarafından erken anlaşılabildiği için bütçe kayıpları azalmış olur.

Medium, yani orta boyuttaki projeler için bahsedilen en uygun model V modelidir. Uzun soluklu projelerde kullanılması uygun değildir, çünkü gereksinimler en başta alınır ve tekrar değiştirilmesi zordur. Test aşamaları ürün geliştirmenin sonunda yapıldığı için risklere açıktır. Çok küçük projelerde de kullanılması gereksizdir, çünkü gereğinden fazla bütçe ayırılacaktır. Küçük çaplı projeler için daha uygun olan modeller şelale modeli ve kodla-düzelt modeli olacaktır. Şelale modelinin kullanımı çoğu proje için kullanışsız olsa da küçük projelerde tercih edilebilir. Koda-düzelt modeli ise büyük projelerde kullanıldığında çok fazla kaynak tüketimine sebep olacağı için küçük projelerde tercih edilebilir.

**5. Çevik Yazılım Geliştirme (Agile)**

Yazılım geliştirme sıkıntılı bir süreçtir. Beklenti ve gereksinim değişikliklerine karşılık verilememesi, sürümlerin zamanında çıkarılamaması, üründeki hataların geç fark edilmesi gibi sorunlar ortaya her zaman çıkabilir. Bu sorunların aşılması için yapılan çalışmalar sonucunda “çevik (agile)” olarak isimlendirilen metotlar geliştirilmiştir. Bu metotlar piyasaya daha hızlı ürün sunabilme, değişen istekleri çabuk yanıtlayabilme, daha kısa sürede müşteri hizmetine ürün sunmayı amaçlar. Çevik yazılım geliştirme metotları daha yüksek verimliliğe sahip, hata oranı düşük, esnek, hızlı ve ucuz çözümler sağlarlar. Çevik yazılım geliştirme metodolojisinde projenin boyutu ne olursa olsun, proje ve geliştirme küçük parçalara, küçük yinelemelere ayrılır ve her yineleme başlı başına birer projeymiş gibi geliştirilir. Her yenilemenin sonunda müşteriye projenin gerçekleştirilen kısmına dair bilgi verilir. Her bir yineleme yaklaşık 2-4 hafta sürer. Yineleme sonlarında müşteriye teslim edilmiş olan çalışan sistem, müşteri memnuniyetini arttırmaktadır. Çevik geliştirmenin hızlı olmasının temel sebeplerinden biri de geliştirme ekibinin sürekli iletişim halinde olmasıdır. Ayrıca projenin küçük parçalardan oluşması geriye dönük hata düzeltmelerini de kolaylaştırır.

**5.1. Çevik yazılım geliştirme prensipleri**

Çevik yazılım geliştirmenin en büyük önceliği müşteriye sürekli ve hızlı bir şekilde kullanılabilir bir yazılım teslimi yapmaktır. Geliştirmenin ilerleyen aşamalarında bile gereksinim değişikliklerine yanıt verilebilir. Çünkü her bir gereksinim bir yineleme içerisinde bir projeymiş gibi incelenip geliştirilebilir. Esnektir. Kısa zaman aralıklarında kaliteli ürün teslimleri gerçekleştirilir. Tüm ekip elemanları birebir iletişim halindedirler ve birlikte çalışırlar. Bu kadar iletişim gerektiren bir metotta ekip içerisindeki kişilerin motivasyonlarının yüksek olması gereklidir. Kaliteli yazılımlar oluşturmadaki temel gereksinimlerden biri ekip motivasyonudur denebilir. Ekip içi kaliteli bilgi akışı ve iyi bir iletişim şarttır. Çevik yazılım geliştirme süreçleri sürdürülebilir geliştirmeyi destekler. Sağlam tasarım, çevikliği arttıracaktır. Ekip düzenli aralıklarla toplanıp yöntemlerini gözden geçirmeli ve verimliliği iyileştirmek için gerekli şeyleri belirlemelidir. Çevik yazılım geliştirmenin avantajları arasında kısa sürede sağlanan müşteri memnuniyeti, yazılım ekibinin motivasyonunu yüksek seviyede tutması, artan üretkenlik, yazılım kalitesi, ve düşen maliyetler sıralanabilir. Ayrıca yazılım projelerinin başarısı yaklaşık %55 artmaktadır.

**5.2. Yaygın Çevik Yazılım Metodolojileri**

8 farklı çevik yazılım geliştirme yönteminden bahsedilecektir. Bunlar “Extreme Programming (XP), SCRUM, Agile Unified Process, Feature-Driven Development (FDD), Test-Driven Development (TDD), LEAN Development, Dynamic System Development Methodology (DSDM), Microsoft Solution Framework (MSF)” olarak sıralanabilir. Bunlardan en çok kullanılan XP ve SCRUM’dan bahsedilecektir. XP; kolay, grup içi iletişime önem veren, geri dönüşlerin daha fazla olmasına imkan sunan bir çevik yazılım geliştirmemetodolojisidir. XP’nin 4 temel değeri vardır: İletişim, basitlik, geri bildirim ve cesaret. Projenin başarıya ulaşabilmesi için ekibin iyi iletişimde olması gerekmektedir. XP’de yüz yüze iletişim tercih edilir. Basitlik, zorunlu olan işlerin yapılmasıdır. Kulağa kolay gelse bile basitliğin sağlanması zordur. En basit çözüm belirlenip uygulanmalıdır. Karmaşık sistem ve çözümler XP mantığına aykırıdır. Geri bildirimler sayesinde ortaya çıkabilecek hatalar ortadan kaldırılır. Müşteriye yapılan ve alınan geri dönüşler önemlidir ve müşteri ekibin bir parçasıdır. Cesaret; projelerin üzerine yılmadan gidilmesi geliştirme aşamasında çok önemlidir. XP, başarısızlıktan korkmamayı ve onun üzerine gitmeyi hedefler.

**5.2.1. Scrum nedir? Neden popüler?**

Scrum, tekrar eden ve aşamalı bir proje yönetim yaklaşımıdır. Sadece yazılım geliştirmede değil, her yerde kullanıp uygulanabilir. Karmaşık yazılım projelerini küçük birimlere bölerek yaklaşır. Karmaşık projelerin adım adım geliştirmesini yapan ekipler için uygundur. Müşteri gereksinimlerindeki değişikliklere yanıt vermede yeterli ve iyidir. Gözleme dayalı bir yaklaşım olarak tanımlanır. Gereksinimlerin kolay tanımlanamadığı ve karmaşık durumların oluşma olasılığının yüksek olduğu projelerde kullanıma uygundur. Bu metodolojide bir yinelemenin tamamlanması en fazla 30 gün sürmekte ve günlük toplantılarla yapılan işlerin takibi ve sürekliliği sağlanmaktadır. Scrum süreci projeyi mümkün olan en fazla ve en küçük parçalara böler. Bu süreç ile zamanın verimli kullanımı en yüksek seviyede tutulmaya çalışılır. Proje ekibinin iletişimi ve işbirliği, uyumu ve verimliliği en üst seviyelere çıkar. Roller, toplantılar ve araçlar olmak üzere 3 temel kavram barındırır. Roller kendi içerisinde ürün sahibi, scrum yöneticisi ve scrum takımı olarak üçe ayrılır. Ürün sahibi ürün değerini yükseltmeyi hedefler. Scrum takımı 5-9 kişiik bir ekiptir ve tek bir hedefe, projeyi geliştirmeye odaklanmıştır. Scrum yöneticisi ise bu takımın scrum temel ilkelerine bağlı kaldığından emin olur ve takımı yönetir. Toplantılar kendi içerisinde sprint planlama, sprint gözden geçirme ve günlük toplantılar olmak üzere üçe ayrılır. Sprint planlama toplantısında geniş bir gereksinim analizi yapılır, takımlar oluşturulur. Her sprint başlangıcında planlama toplantısı yapılır. Sprint gözden geçirme toplantısında, önceki toplantıda yapılan gereksinim analizi sonucu oluşturulan liste gözden geçirilir ve “koşu gereksinim listesi” oluşturulur. Bu, scrum’ın anahtar uygulamasıdır. Günlük toplantılar sprint başladıktan sonra projenin ilerleyişi ve ekibin karşılaştığı engelleri görmek için yapılır. Araçlar kendi içerisinde ürün gereksinim dokümanı, sprint dokümanı ve sprint kalan zaman grafiği olarak üçe ayrılmıştır. Ürün gereksinim dokümanı. Proje boyunca yapılacakların basit bir listesidir. Gereksinimler değiştikçe bu doküman da değiştirilebilir. Canlıdır. Sprint dokümanı ise sprint sonunda elde edilecek ürünün gereksinimleridir. Sadece ekip tarafından değiştirilebilir. Sprint kalan zaman grafiği, normalde ne kadar iş yapılması gerektiği ve her gün ne kadar iş yapıldığını karşılaştırır. Verimliliği ölçecektir.

Scrum hafif derecede bir proje yönetim sürecidir, farklı boyuttaki projelerde uygulanabilir. Scrum'ın popüler olmasındaki temel özelliklerinden birkaçı şu şekilde sıralanabilir;

* Küçük geliştirme ekiplerini sağlıklı iletişim ve bilgi akışına teşvik eder.
* Değişen kullanıcı gereksinimleri için rahat uyarlanabilme şansı sağlar.
* Sık sık geliştirilen ürün sürümlerinde, test edilmiş ve onaylanmış olanlar dokümante edilebilir.
* Yapılacak ana projenin küçük parçalara ayrılması ve mümkün olduğunca birbirinden bağımsız küçük, yinelemelerle tamamlanabilecek iş parçalarının sağladığı kolaylık.
* Her yineleme sonunda teslim edilen bir proje geri bildirimi, sürekli yapılan bu geri bildirimler sayesinde mali ve teknik kaynakların kullanımının çok daha verimli olması.

**6. KAYNAKÇA**

* [**https://systemanalize.wordpress.com/gelistirme-metotolari/**](https://systemanalize.wordpress.com/gelistirme-metotolari/)
* [**https://medium.com/architectural-patterns/yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-modelleri-62915545c51e**](https://medium.com/architectural-patterns/yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-modelleri-62915545c51e)
* [**https://medium.com/@brfn.kcr26/yazilim-geli%CC%87%C5%9Fti%CC%87rme-ve-s%C3%BCre%C3%A7-modelleri%CC%87-2131ea5f09b2#:~:text=1.Kodla%20Ve%20D%C3%BCzelt%20Ya%C5%9Fam,kadar%20kodlama%20yap%C4%B1larak%20devam%20edilir.&text=%C3%BC%20%C3%87ok%20k%C3%BC%C3%A7%C3%BCk%20projelerde%20ya,%C3%BC%20Program%20a%C5%9Famalar%C4%B1%20%C3%A7abuk%20ge%C3%A7ilir**](https://medium.com/@brfn.kcr26/yazilim-geli%CC%87%C5%9Fti%CC%87rme-ve-s%C3%BCre%C3%A7-modelleri%CC%87-2131ea5f09b2#:~:text=1.Kodla%20Ve%20D%C3%BCzelt%20Ya%C5%9Fam,kadar%20kodlama%20yap%C4%B1larak%20devam%20edilir.&text=%C3%BC%20%C3%87ok%20k%C3%BC%C3%A7%C3%BCk%20projelerde%20ya,%C3%BC%20Program%20a%C5%9Famalar%C4%B1%20%C3%A7abuk%20ge%C3%A7ilir)**.**
* [**https://www.tutorialspoint.com/sdlc/sdlc\_waterfall\_model.htm#:~:text=The%20Waterfall%20model%20is%20the,the%20previous%20phase%20is%20complete**](https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm#:~:text=The%20Waterfall%20model%20is%20the,the%20previous%20phase%20is%20complete)**.**
* [**https://medium.com/@esra\_/v-model-nedir-34d5103d183a**](https://medium.com/@esra_/v-model-nedir-34d5103d183a)
* [**https://tr.wikipedia.org/wiki/V-Model\_(Yaz%C4%B1l%C4%B1m\_geli%C5%9Ftirme)**](https://tr.wikipedia.org/wiki/V-Model_(Yaz%C4%B1l%C4%B1m_geli%C5%9Ftirme))