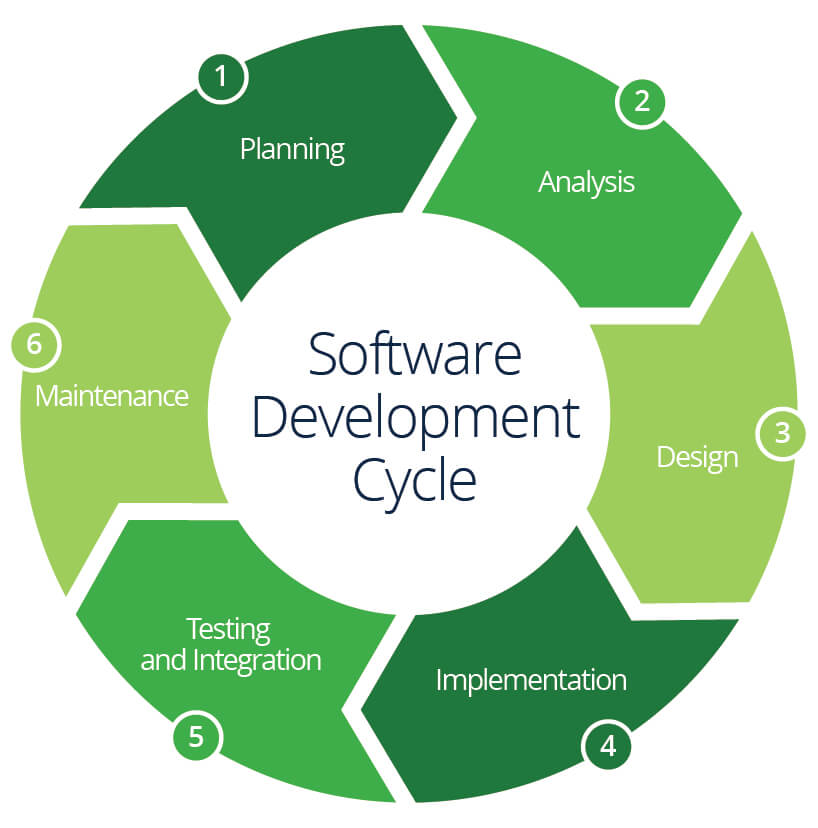
**Ragıp Kaan Söylev-200601057**

**YAZILIM YAŞAM DÖNGÜ MODELLERİ**



**YAZILIM YAŞAM DÖNGÜSÜ NEDİR?**

Birçok alanda proje yaparken izlenmesi gereken yollar vardır. Yazılım projelerinde de bu yolların genel adına “Yazılım Yaşam Döngüsü” adı verilir. Yazılım yaşam döngüleri birçok çeşittedir ancak yine de bu yaşam döngülerinin bazı temel aşamaları vardır. Planlama, analiz, tasarım, gerçekleştirme, bakım aşamaları bu temel aşamaları oluşturur. Yazılımın yaşam süresinde bu aşamalar üzerinden tekrar tekrar geçilebilir yani sürekli bir dönüşüm vardır.

Planlama: Yazılım yaşam döngülerinin başlangıcını oluşturur. Projenin nasıl yapılacağı, nelere ihtiyaç duyulduğu hakkında planların yapıldığı aşamadır.

Analiz: Yazılım projesinin işlevleri ve gereksinimleri belirlenir. Bu yaşamada yazılım ekibi, iş analisti, müşteri arasındaki iletişim çok önemlidir. Bu aşamada iyi yapılamayan analiz daha sonra çok fazla masrafa neden olabilir.

Tasarım: Yazılım projesinin özellikleri, görünümü belirlenir. “Üst düzey tasarım” ve “Düşük düzey tasarım” olmak üzere iki farklı tasarım biçimi vardır. Üst düzey tasarımda modüllerin isimleri yazılır, modüllerin işlevleri hakkında taslaklar oluşturulur. Düşük düzey tasarımda ise arayüzün tam tanımlanması yapılır, modüllerin girdi ve çıktıları tanımlanır.

Gerçekleştirme: Projenin kodlama ve test işlemleri bu aşamada yapılır. Test aşaması hata oranımızı düşürüp ileride çıkacak sorunlara engel olması açısından çok önemlidir.

Bakım: Proje teslim edildikten sonra ortaya çıkan hataların düzeltilmesi, yeni özelliklerin eklenmesi, eski özelliklerin güncellenmesi veya çıkarılması işlemlerinin yapıldığı aşamadır.

**YAZILIM YAŞAM DÖNGÜ MODELLERİ**

**Gelişigüzel Model**

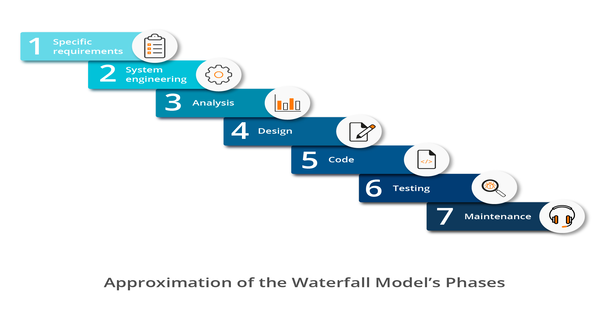
1960’lı yıllarda kullanılan bu model günümüze göre çok ilkel kaldığı için kullanımı çok nadir hale gelmiştir. Baktığımız zaman sistem ve yöntem barındırmadığı için de model olarak adlandırmak da yanlış olacaktır. Kişiler bu modelde yazılımı kendilerine göre oluşturduğu için kodların yeniden okunabilirliği, bakımı zordur.

**Barok Modeli**

1970’li yıllarda kullanılan bu modelde yaşam döngüsünün temel adımları tek tek sıra halinde döngü olmaksızın uygulanır. Dokümantasyon aşaması gerçekleştirme aşamasından sonra yapılır. Bu model de günümüz modellerine göre çağ dışı kaldığı için çok nadir kullanılmaktadır.

**Şelale Yaşam-Döngü Modeli**

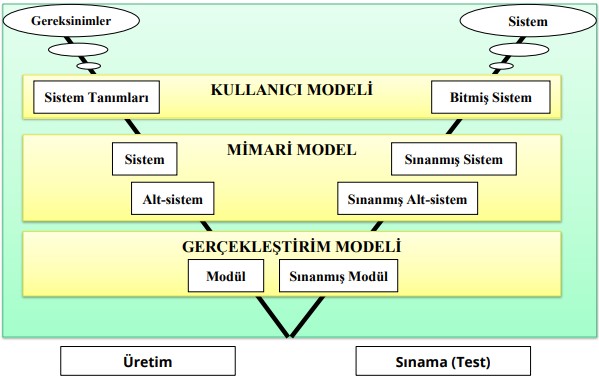
Bu model geçmişte çok popülerdir ve günümüz modellerine temel olacak nitelikte özellikler taşır. Bu modelde yazılımın temel aşamaları en az bir kez tekrar edilir. Model çok iyi tanımlanmış ve kısa sürede bitebilecek yazılım projelerinde kullanılır. Dokümantasyon aşaması ayrıca yapılmadan ana işlemler devam ederken yapılır. Dokümantasyon her aşamada yapılır, dokümantasyonu yapılmayan aşama yapıldı olarak kabul edilmez.



Bu modelin aşamalarında kullanıcı yer almadığı için kullanıcılardan gelecek geri dönüşler artar ve bu durum projenin maliyetini arttırabilir. Şelale modelinde kodlama aşaması çok az yer aldığı, dokümantasyon ve diğer işlemlere yoğunlaşıldığı için yazılımcılar motivasyon kaybına uğrayabilirler ve bu durum da projenin süresinin artmasına dolayısıyla maliyetlerin yükselmesine neden olur.

**V Süreç Modeli**

Projedeki tanımların belirgin olduğu, belirsizliklerin az olduğu projelerde kullanılır. Model kullanıcı katkısına çok önem verir.



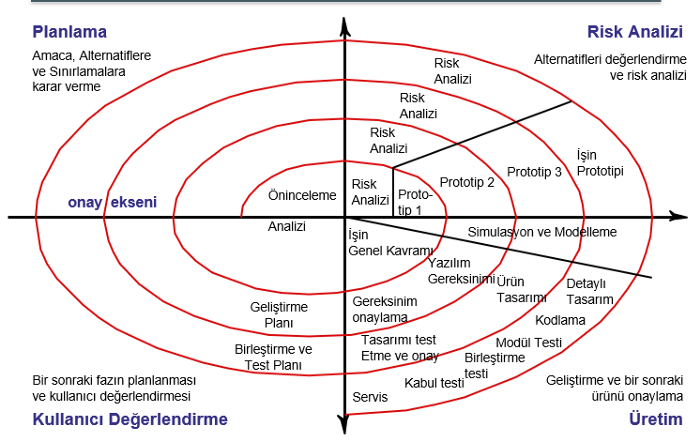
Sol taraf üretim, sağ taraf ise sınama işlemlerini gösterir. Bu modelin üç temel çıktısı vardır: Kullanıcı Modeli, Mimari Model, Gerçekleştirim Modeli. Bu aşamalarda tekrar bulunmaması projenin ileri safhalarında sorunlara neden olabilir.

-Kullanıcı Modeli: Sistemin gereksinimleri ve kullanıcının ihtiyaçları belirlenir. Bu veriler ışığında kullanıcı gereksinim dokümanı oluşturulur. Kullanıcının sistemi nasıl istediğine ilişkin testler de bu aşamada yapılır.

-Mimari Model: Projenin tasarımı, sistem ve alt sisteminin test işlemlerinin yapıldığı aşamadır.

-Gerçekleştirim Modeli: Kodlamanın yapıldığı ve bu kodların test edildiği aşamadır.

**Helezonik (Spiral) Model**



Bu modelde en göze çarpan özellik her fazda risk analizi yapılması ve prototip oluşturulmasıdır. Oluşturulan prototipler sayesinde kullanıcıdan her fazda geri dönüş alınır ve bu sayede gelecek aşamalarda çıkacak sorunlar azalır. Yazılımcılar projenin erken safhalarında kodlamaya ve teste başladıkları için bu durum yazılımcıların verimini arttırır. Spiral modelin olumlu özellikleri olduğu kadar olumsuz özellikleri de vardır. Örneğin spiral model küçük projeler için çok pahalıdır ve karmaşık bir sistem olduğu için uzun sürer bu da ayrıca maliyeti arttırır. Spiral modelin dört aşaması vardır:

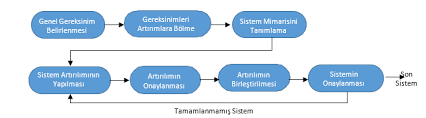
-Planlama: Üretilen ara ürünler için planlamalar yapılır ve önceki adımdaki ara ürünle entegre edilir.

-Risk Analizi: Risklerin araştırıldığı ve belirlendiği aşamadır.

-Üretim: Ara ürünün üretilmesi

-Kullanıcı Değerlendirmesi: Kullanıcının ara ürünü test etmesi sonucu verdiği geri dönüşler.

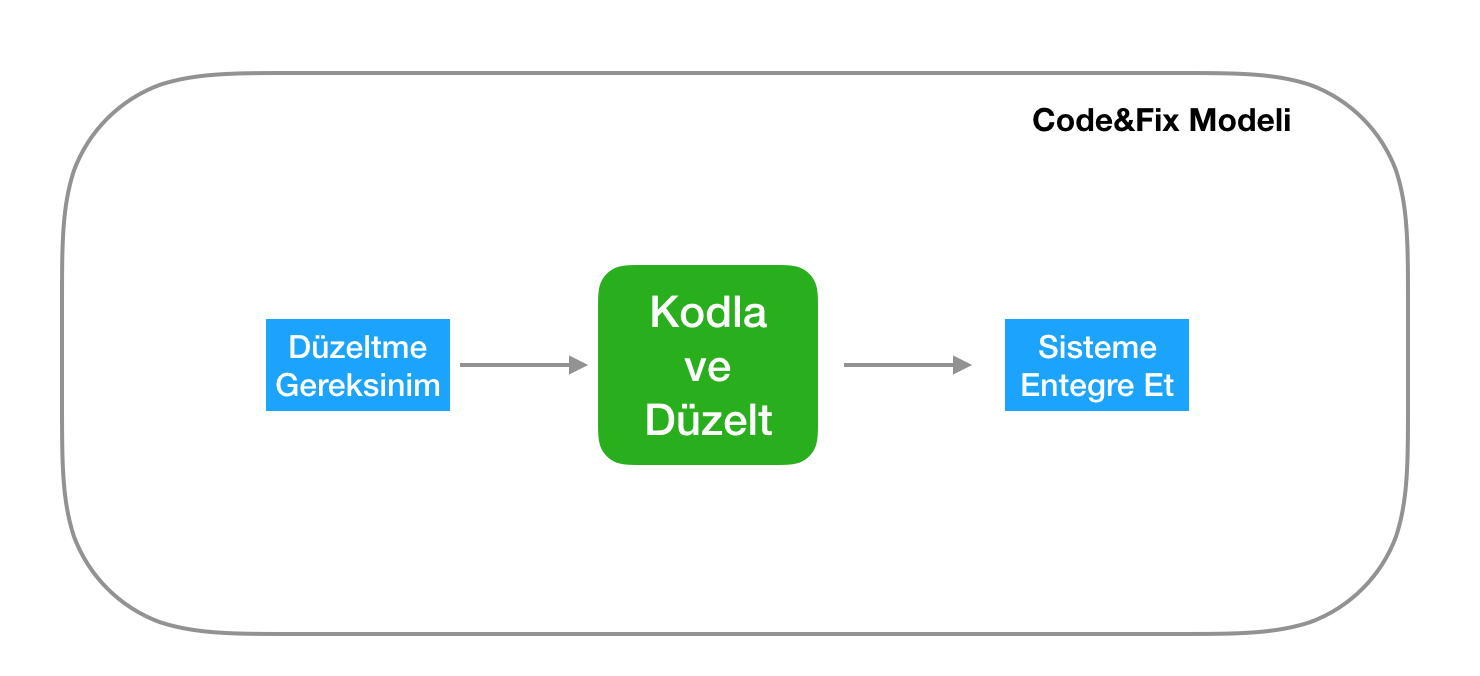
**Artımsal Geliştirme Süreç Modeli**



Bu modelde tek bir teslim işlevi görülmez, bunun yerine projenin sürekli geliştirilir ve geliştirmeler yapıldıkça yeni yazılım sürümü teslim edilir. Gereksinimler önceliklerine göre parçalarına bölünür ve en önemli gereksinimler erken teslimlerde tamamlanırlar. Bir parça geliştirilmeye başlandığında bu parçaya yeni gereksinimler eklenmez.

Yapılan her teslimde bir önceki sürümün üzerine bir şeyler katılarak projenin işlevselliğinin arttırılması sağlanır. Artımsal geliştirme modeli ile yapılan projeler uzun sürede tamamlanırlar ve bu projelerin eksik işlevsellikle çalışabilmesi gerekmektedir. Bu modelde proje üzerine sürekli iyileştirmeler yapıldığı için projenin başarısız olma olasılığı azalır. Projenin öncelikli gereksinimleri diğer gereksinimlere göre daha çok test edilir.

**Kodla ve Düzelt Yaşam Döngü Modeli**



Kodla ve düzelt modeli küçük projeler için kullanılır. Yazılım geliştirmek çok kolaydır ancak bakım yapmak çok zordur ve masraflıdır bu sebeple büyük çaplı projelerde bu model kullanılmaz.

**Çevik Modeller**

Çevik yazılım modelleri çok kısa sürelerde müşteriye yazılım ürününü sunmayı, kısa sürelerde daha çok yazılım projesi üretebilmeyi sağlar. Bu modellerde yazılım ekibi, analistler, testçiler, müşteri gibi kişiler arasındaki ilişki çok önemlidir. Çevik modellerde sürekli yineleme yapılması yazılımdaki hata oranını düşürür ayrıca müşteri eline çalışan yazılım geçtiği için memnun olur.

En Yaygın Uygulanan Çevik Metodolojiler: 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)

**Extreme Programming (XP)**

1990 yılında Kent Back tarafından ortaya çıkarılan bir yazılım geliştirme metodudur. Bu metodun önem verdiği dört madde vardır: İletişim, Basitlik, Geri Bildirim, Cesaret.

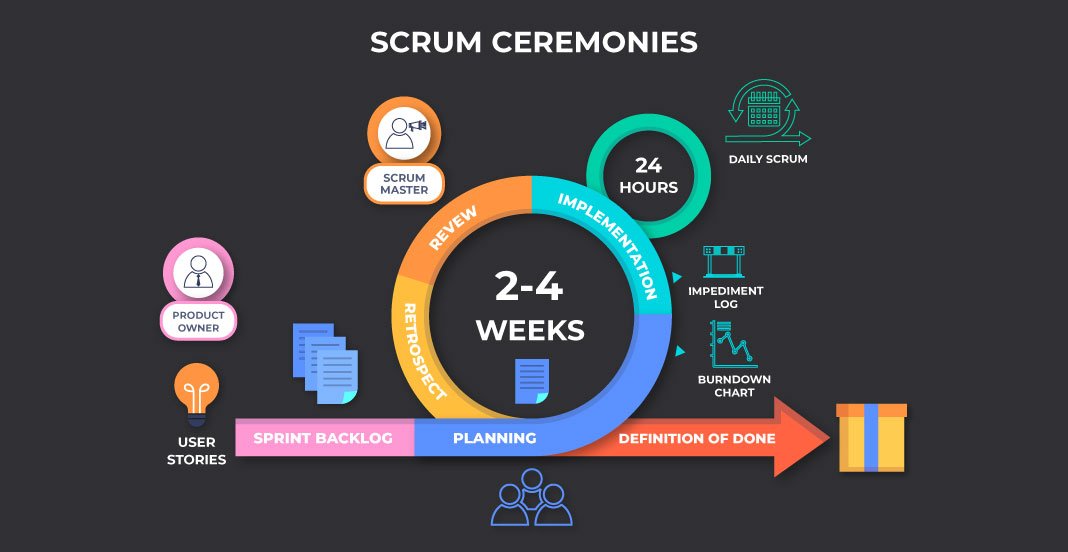
-İletişim: Yazılım geliştirmede iletişim çok büyük rol oynar. XP metotlarında da bu duruma çok önem verilir, bu sebeple XP kullanılan projelerde iletişimin yüz yüze olması gerekmektedir çünkü böyle yapılırsa herhangi bir sorun hemen çözülebilir ve yazılım geliştirme hızı yavaşlamamış olur.

-Basitlik: XP metotlarında verimliliğe ve hıza çok önem verildiği için projenin gereksinimlerinin karmaşık yöntemlerle değil basit yöntemlerle yapılması, ortaya çıkan sorunların basit düşüncelerle çözülmesi sağlanır.

-Geri Bildirim: Müşteriyle iletişim için geri bildirim çok önemlidir. Bu sebeple XP uygulanan projelerde yazılım ekibi ve müşteri belirli zaman buluşurlar ve yazılımın gelinen noktadaki eksiklerini verimini tartışırlar. Bu tartışmalardan elde edilen veriler projenin ilerlemesi için faydalı olur.

-Cesaret: XP metotlarında cesaretli olmak, gerektiği yerde radikal kararlar alabilmek önemlidir. Projenin başarısız olacağı düşüncesiyle ilerlenmemeli, önümüze çıkan her engeli aşabiliriz düşüncesiyle ilerlenmelidir. Projede başarısız olunsa bile yeniden denemeli, eski hatalardan ders çıkarılmalıdır.

**SCRUM**



Büyük yazılım projelerinde kullanılır, projenin “sprint” adı verilen parçalara bölünüp geliştirilmesi sağlanır. Bir sprintin 30 günden fazla sürmemesi gerekir. Ekip içi iletişim önemlidir, bunun için günlük olarak on beş dakikalık toplantılar yapılır. Scrum’da bulunan üç kavram scrumın temelini oluşturur:

Roller: Ürün sahibi, projenin mali yönüyle ve eline geçen ürünle ilgilenir. Scrum yöneticisi, scrum takımının scruma yoğunlaşmasını, kuralların ihlal edilmemesini, scruma uyulmasını sağlar. Scrum takımı, disiplinli bir şekilde, yazılım geliştirmeye odaklanan 5-9 kişilik yazılım takımıdır.

Toplantılar: Her sprintten önce sprintin gereksinimlerini, ortaya çıkabilecek sorunları belirleyebilmek, sprint takımlarını oluşturmak, sprintin maliyetini hesaplamak amacıyla sprint planlama toplantısı yapılır. Bu toplantıdan sonra takımlar bu sprintte hangi gereksinimleri yapabileceğini belirler ve bunu ürün sahibine sunarlar. Günlük sprint toplantıları, sprint başladıktan sonra günlük olarak yapılan, ortalama on beş dakika süren toplantılardır. Takım üyelerini bir önceki gün neler yaptıklarını, ne gibi sorunlarla karşılaştıklarını, o gün içinde neler yapacaklarını tartışırlar.

Bileşenler/Araçlar: Ürün gereksinim dokümanı, üründe olması gereken gereksinimlerin yazıldığı bir dokümandır. Üzerinde sürekli güncelleştirmeler yapılabilir. Oluşturulmasında kullanıcıdan alınan geri dönüşler önemlidir. Sprint dokümanı, ürün gereksinim dokümanından sprint için alınmış olan gereksinimlerden oluşur. Sprintte yapılacaklar bu dokümana göre oluşturulur, sadece sprint takımı içindekiler bu dokümanı değiştirebilir. Sprint kalan zaman grafiği, scrum takım üyelerinin günlük olarak ne kadar iş yaptıklarının bilgisini girdikleri, sprintin verimliliğini ölçek için kullanılan bir grafiktir.

Scrum metodu günümüzde en çok kullanılan metottur. Bunun sebebleri: büyük projelerdeki karmaşıklığı azaltması, ekip içi ve müşteri-ekip arasındaki ilişkinin çok güçlü olması, sorunların hemen çözülebilmesi, zaman ve para kullanımını azaltması, kullanıcı geri dönüşlerine fazla yer almasıdır.

**Modellerin Karşılaştırılması**

Gelişigüzel model ve barok modeller günümüzde çok önem verilen dokümantasyon, test, iletişim gibi kavramları karşılamadı için kullanılamaz duruma gelmişlerdir.

Şelale modeli, eski dönemlerde günümüze göre daha küçük projeler üretildiği için kullanılması çok fazla yaygındı ancak gelişigüzel ve barok modellerde olduğu gibi iletişimin düşük olmasından dolayı günümüzde kullanılamazlar.

V Modeli, belirsizliklerin az olması, yapılan tanımların tam olması V modeline büyük avantajlar sağlar ancak modeller arası tekrarların olmaması bu modeli dezavantajlı duruma düşürür.

Spiral model, protitip oluşturulmasını, risk analizini, risk azaltımını ön plana alır. Bu özellikleri barındırmasına rağmen karmaşık olması, uzun sürmesi, sürekli prototip çıkarılmak zorunda olunması bu modeli küçük projelerden daha çok büyük projeler için kullanılabilir yapar.

Artımsal geliştirme modelinde birden fazla teslim yapılarak sürümler oluşturulması, müşteriden geri dönüt alınması ve ona göre geliştirmeler yapılması açısından avantajlıdır ancak eksik işlevsellikle çalışabilecek yazılımlarda kullanabileceği için bu durum modelin kullanım alanını azaltır.

Kodla ve düzelt modeli teslimden sonraki bakım safhasını yapmak çok zor ve maliyetlidir bu sebeple büyük projelerde kullanılamaz. Genelde tek bir kişinin kendi metotlarıyla, kendi istediği gibi yapılacak projelerde kullanılır, dokümantasyon yazımı görülmez.

Çevik modellerde takım içi iletişim, müşteri ile iletişim, zaman, kaynak, finans kullanımı çok önemlidir. İletişime çok önem verildiği için projedeki hatalar çok azalır. Bu özellikleriyle birlikte günümüzde oldukça fazla kullanılır.

**Hangi Projede Hangi Modeli Kullanmalıyız?**

-Küçük ve detaylı olmayan projelerde şelale modelini kullanabilir.

-Belirsizliklerin az olduğu, iş tanımlarının iyi yapıldığı projelerde V Modeli kullanılabilir.

-Büyük, uzun zaman alacak, hata payının düşük tutulması gereken projelerde spiral model kullanılabilir.

-Müşterinin sürekli yeni sürümler istediği, eksik işlevle çalışabilecek projelerde artımsal geliştirme modeli kullanılabilir.

-Kişiye özel, küçük, daha sonra bakıma ihtiyaç duyulmayacak projelerde kodla ve düzelt modeli kullanılabilir.

-Zaman ve kaynak kullanımının önemsendiği; iletişimin, hata oranı düşüklüğünün ön planda olduğu, küçük-orta-büyük fark etmeksizin her türlü projede çevik modeller kullanılabilir.

**Yazıda Kullanılan Kaynaklar**

- Doç. Dr. Deniz Kılınç, Bakırçay Üniversitesi Yazılım Mühendisliğine Giriş Dersi 2. ve 3. Hafta Ders Notları

<https://en.wikipedia.org/wiki/Systems_development_life_cycle>

### https://en.wikipedia.org/wiki/Agile\_software\_development <https://www.reddit.com/r/cscareerquestions/comments/3ff1ns/what_exactly_is_a_software_development_life_cycle/>

### <https://www.reddit.com/r/explainlikeimfive/comments/8epp98/eli5_what_is_in_programming_the_scrum_and_agile/>

<https://www.youtube.com/watch?v=G-6qDY8UltU>

<http://rationalizedthoughts.blogspot.com/2005/11/what-are-requirements-sdlc.html>

<https://stackoverflow.com/questions/2755167/what-is-the-software-development-lifecycle>

<https://softwareengineering.stackexchange.com/questions/15928/what-is-agile-methodology>

<https://content.intland.com/blog/agile/scrum/why-is-scrum-so-popular-why-is-scrum-so-successful#:~:text=In%20conclusion%2C%20Scrum%20is%20the,done%20within%20a%20teamwork%20environment>.