**Yazılım Yaşam Döngüsü Nedir?**

"Software Development Life Cycle" kısaca SDLC, bir yazılım projesinin en iyi şekilde yürütülebilmesi ve zamanında geliştirilebilmesi,kullanıcının gereksinimlerinin tamamı karşılayabilmesi ve sürekliliğini koruyabilmesi için geliştirilen metedolojiler bütünüdür.Müşterilerin genişleyen istekleri ve gereksinimleri , yazılım teknolojilerindeki değişiklerden dolayı süreç tek yönlü olmayıp geriye dönüşlerin olduğu bir döngü haline gelir.Bu döngüler ile sorunlar tekrar tekrar çözülerek ve projeye yeni fonksiyonlar ekleyerek daha kolay ve eksiksiz,planlı bir şekilde ilerlenilmesi sağlanır.Bu sayede zaman, maaliyet ve performanstan kazanç elde edilir.

SDLC aşağıdaki aşamalardan oluşur:

**1.Planlama:**Projenin ilk ve en temel aşamasıdır.Bu aşama başarılı bir proje geliştirebilmek için projenin önceden tüm resminin ortaya dökülmesidir.Burada proje ekibi "Ne istiyoruz?" sorusuna yanıt aramalıdır.Başka bir deyişle projenin fizibilitesinin çıkartıldığı ve projeyi en verimli ve en başarılı bir şekilde nasıl faaliyete gecirilebileceğinin belirlenen aşamadır.

**-Fizibilite Çalışması**

İngilizce kökenli, feasible kelimesinden türetilen “Fizibilite” kelimesi mümkün,uygulanabilir,yapılabilir,olası anlamlarına gelmektedir. “Fizibilite nedir?”sorusuna ise proje fikrinin gerçekliğe uygunluğunun saptanmasıdır diyebiliriz.Fizibilite aşamaları,temel prensip olarak pazar araştırması yapılarak teknik ve finansal konulardaki değerlendirmelerle bir rapor halinde sunularak başlangıç aşaması için bir yol gösterici niteliğinde olur.Bu şekilde projenin ölü bir ürün olmasının önüne geçilir.

**2.Analiz:**Bu aşama artık sistem gereksinimlerinin ve işlevinin müşteri tarafından onaylanıp kesinleşmesi sonucunda bu verilerin dökümante edilmesidir.  
**3.Tasarım:**Bu aşama "Bu ürünü nasıl inşa edeceğiz?” sorusuna yanıt vermelidir.Burada ürünün yol haritası çıkarılır.Ekipteki herkesin amaç ve görevlerinin daha net bir şekilde görülmesini sağlar.  
Tasarımda en önemli ve en çok kullanılan tekniklerden bir tanesi de Soyutlama(Abstraction) tekniğidir.Soyutlama ile bir kavramın bilgi içeriğini aza indirgemeyi amaçlayarak asıl gerekli olan bilgiye daha kolay ulaşmak hedeflenir.Yani bu teknikle problemin çözümü için en önemli kısıma odaklanarak geri kalan kısım görmezden gelinir.  
**4.Kodlama:**Uzun bir süreci temsil eder.Müşteriye sunulacak ürünün programlama aşamasıdır.burada soruları cevaplamaktan ziyade çıktılar üretmeyi amaçlayan ürünün inşa edildiği kısımdır.  
**5.Test:**Yazılan ürünler her zaman müşterilere hizmet eder,bu nedenle verilen hizmetin kalitesi son derece önemlidir.Örneğin;girdiğimiz internet sayfasının açılmadığı,akıllı cep telefonlarımızın hata verdiği,kullandığımız uygulamalar,oyunlar vs yuklenmediği durumlar can sıkıcı olabiliyor. Müşterinin aklında ürünümüz iyi bir deneyim olarak kalabilmesi için ürünün sorunsuz ve eksiksiz bir şekilde çalışması gerekir.Üretilen her şeyi kontrol etmek zor olduğu gibi gözden kaçırılan ayrıntılarda olabilir.Bu sebeple erken test yaklaşımı ile hata yapma oranımızı ve oluşacak maaliyetleri düşürerek ürün kalitesini arttırırız.  
Yazılım testi aşamasıyla  
-yüksek ürün maliyetinden ve mali zarardan korunulunur  
-üründe oluşabilecek güvenlik açıklarını minimuma indirilir  
-müşteri memnuniyeti ve güvencesi kazanılınır.  
**6.Bakım:**Yazılım ürünün tüm aşamaları tamamlanıp teslim edildikten sonra müşteri geri dönüşleriyle yapılan bakım ve yeni güncellemelerin getirildiği aşamadır.  
  
**Software Development Life Cycle Modelleri**  
Yazılım endüstrisi Türkiye’nin de aralarında bulunduğu gelişmekte olan ülkelerin çoğunda, büyük önem kazanmaya başlamıştır.Zamanında harekete geçip gerekli önemi gösteren ülkelerin ortaya koyduğu sektördeki başarı örnekleri diğer ülkeler için de cezbedici olmaktadır. Ancak benzer başarılara ulaşmak;değişen koşullar,artan rekabet ve endüstrinin gerektirdiği altyapıların hazır olmayışı gibi nedenlerle kolay değildir.Yazılımın üretim süreci bir dizi mühendislik ve proje yönetim sürecini gerektirir.Bu noktada SDLC modellerine ihtiyaç duyulur.Projenin büyüklüğü,karmaşıklığı ve yazılımı üretecek organizasyon ve proje ekibinin yapısı gibi faktörlere göre bir yazılım proje yönetimi modeli seçilir. Bu modeller geleneksel ve çevik yazılım modelleri olarak iki temel kısma ayrılmıştır.

#### **Geleneksel Yazılım Modelleri** **Gelişi Güzel Model** Bu yöntem 1960’lı yıllarda kullanılmıştır.Adından da anlaşılacağı gibi bu model yöntem gerektirmez.Genellikle tek kişiye bağlıdır.Yazılımın izlenebilirliği ve bakımı çok zordur. **Barok Modeli**

1970’li yıllarda ortaya çıkmıştır.Yaşam döngüsü temel adımları doğrusal bir şekilde ele alınır ve geliştirilir. Aşamalar arasında geri dönüşler sağlanmamıştır. Bu modelin ayırıcı özelliği; "Belgeleme" olgusuna ayrı bir önem vermesi, belgelemeyi bir süreç olarak ele almasıdır ve yazılım geliştirme ve test faaliyetleri tamamlandıktan sonra yapılmasını öngörür. Günümüz yazılım geliştirme projelerinde uygulanan bir model değildir.  
**ŞELALE MODELİ(WATERFALL MODEL)**  
Şelale modeli, yazılım projelerinde uygulanacak faaliyetlerin ardışık sıralar şeklinde dizayn edildiği en eski ve en temel modeldir.Her aşamanın sonunda bir çıktı elde edilir ve bu çıktı sıradaki aşamanın girdisini oluşturur.Yaşanacak olası değişimler çıktı-girdi ilişkisinden dolayı büyük maaliyetler doğurabileceğinden safhalar arası atlama ve geri dönüşlere izin verilmez. Sistem ve yazılım gereksinimlerine gerekli her şey en başta belirlenir . Bu nedenle belirsizlik içermeyen tam anlaşılmış ,değişme ihtimali olmayan,basit ve kısa projeler için daha uygundur.  
**V Süreç Modeli**  
Şelale modelinin bir varyasyonudur.V modeli her aşama için uygun bir test aşamasına sahiptir.Her aşama kendi kontrol aşamasıyla eşleştirilerek V şeklinde gösterildiği için bu ismi almıştır. V şeklinin sol ayağı tanımlanama ve üretim süreci sağ ayağı ise test etkinlikleri ile ilgilidir.Gereksinimleri açık ve net olan küçük,orta ve büyük boyutlu projelerde kullanılabilir.V modelinin tercih edildiği alanlar e-devlet uygulamaları ve finansal yazılımlardır.  
**V-modelinin avantajları:**

* Basit ve kullanımı kolaydır.
* Planlama ve test tasarımı gibi test faaliyetleri kodlamadan önce gerçekleştirildiği için proje içerisinde çok zaman kazandırır. Bu nedenle şelale modeline göre daha yüksek başarı şansı vardır.
* Hataların bulunması erken aşamada bulunur.
* Hataların bir sonraki aşamaya geçmesi önlenir.

**V modelinin dezavantajları:**

* Uygulama şekli oldukça katı, kesin kurallara bağlı,disilpinlidir.
* Yazılım şelalede olduğu gibi geliştirme aşamasında geliştirilir, bu nedenle yazılımın erken prototipleri üretilmez.
* Herhangi bir aşamada gereksinimler üzerinde değişiklik olursa, test belgelerinin de diğer belgeler ile birlikte güncellenmelidir.

**Helezonik (Spiral) Model**  
Spiral model,risk odaklı bir yazılım geliştirme süreci modelidir. Amacı riski seviye seviye minimuma indirerek projenin başarılı bir şekilde tamamlanmasını sağlamaktır.Şelale modeli ile yinelemeli modelin birleşimidir.Bu modeli çok sayıda döngüsü olan bobine benzetebiliriz.Spiralin her döngüsü yazılım yaşam döngüsünün bir aşamasını temsil eder.Döngü sayısı her projeye ve ekip yapısına göre değişir. Spiral modelini bir salyongoz kabuğu gibi giderek üstüne büyüyen bir geliştirme sistemi olarak görebiliriz.  
4 ana bölüm içerir.Bunlar planlama,risk yönetimi,üretim ve kullanıcı değerlendirmesidir.  
Spiral modelin bir çok avantajı vardır. Bunlar şöyle sıralanabilir;

* Üretim süreci boyunca, ara ürünü değerlendirirken ve test ederken kullanıcının aktif olması, kullanıcının sistemi erkenden tanımasına yardımcı olur ve ilerde oluşacak istemeyen durumları en aza indirger.
* Bu modelde yöneticiler ve proje sahipleri daha çok karşılaşma olanağı sağlarlar. Böylece projenin izlenimi daha kolay olur.
* Bu modelde yazılım kodlanması ve sınanması daha erken başlar.
* Yinelemeli bir yaklaşım vardır. Buda hataları daha aza indirger.

#### **Artırımsal(Incremental) Geliştirme Modeli**

Artırımsal Geliştirme modelinde proje küçük parçalara ayrılır ve bu parçalar kullanıcının önceliğine göre sıralanır.Sıralanan bu parçalar tamamlandığında sırasıyla birer ara ürün oluştulurmuş olunur.Ara ürünler her seferinde bir öncekinin üstüne yeni fonksiyonlar katarak çıkartılır.Bu model uzun zaman alabilecek ve ürünün eksik işlevsellikle çalışabileceği türdeki yazılım projeleri için uygundur.

**Çevik Modeller**  
Çevik modeller;zaman içerisinde yazılım sektörünün büyümesi ve daha karmaşık bir hal almasıyla müşterinin büyük resmi göremeyip gereksinimlerini tam olarak ortaya koymaması sürekli ve hızla değişen yenilikleri yazılım projelerine entegre edilemesinde zorluk yaşanması gibi problemlerden ortaya çıkmıştır.Günümüzde popülerliği güçlü bir konumdadır.Türkçede kelime anlamı çevik olan “Agile” kelimesinin kökeni,İngilizcede çeviklik anlamına gelen Agility kelimesidir.Agility yani çeviklik, değişen koşullara hızlı ve kolay cevap verme kabiliyetidir.Dolayısıyla çevik modelin en temel amacı ürünü en kısa sürede piyasaya çıkarabilmek,değişen isteklere hızlı yanıt verebilmektir.Kapsadığı değerler, prensipler ve pratikler sayesinde geleneksel modellemelere metotlarına göre daha esnek ve kullanışlıdır.4 ana temel değer vardır:

1)Müşteri ile iş birliği: Müşterinin, tüm süreçler içerisinde yer alması istenir. Müşteri-danışman ilişkisi yerine, bir ekip dayanışması ile el ele ilerlenir.  
2) Bir plana bağlı kalmak yerine değişimlere hakim ve cevap verebilir nitelikte olmak: Günümüzün çok hızlı değişen koşullarına hızlı cevap verebilmek, hem bireyler hem de tüzel kişilikler için bugünün en kıymetli yetkinliklerden. Bir yazılım projesinde, yaklaşık bir sene sonra kullanılacak bir sistemin özelliklerini bugünkü ihtiyaçlarımıza göre belirlemek ve bu kapsamda değişime açık olmamak riskli olacaktır.  
3) Kapsamlı dokümantasyondan ziyade çalışan yazılım:Tasarım dökümantasyonu üzerinde vakit öldürmekten ziyade elimizde sürekli çalışan bir yazılım olmasını öncelikli kılar.  
4)Bireysel Etkileşimler  
  
En yaygın çevik yazılım metodolojileri ;” 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 bilinen çevik metodolojilerdir.”  
  
**Extreme Programming (XP)**  
Kent Beck ve arkadaşları tarafından 1996 yılında Chrysler firmasında yapılan bir proje bünyesinde oluşmuştur. İçerdiği basit ama bir o kadar etkili yöntemlerle yazılım sektöründe yeni bir rüzgarın esmesini sağlamıştır.  
  
XP modeli müşteri gereksinimlerini merkezi bir konumda tutar.Yazılım esnasında çıkan bir değişikliğe kolayca uyum sağlayabilir.Projede çevik olabilmek için en az yükle çıkılması gerekir.Bu yüzden projelerde formalite ve bürokrasinin en az seviyeye indirilmesini söyler ve proje öncesi geniş anlamda tasarım ve dokümentasyon oluşturulmasına izin vermez.  
XP dört temel değer üzerine kuruludur: Basitlik (Simplicity), İletişim (Communication), Geridönüm (Feedback) ve Cesaret (Courage).  
Bu değerlere bir açıklık getirelim.XP en basit yöntemler aracılığıyla sonuca ulaşmak ister, çünkü sadece bu şekilde hızlı ve düşük maliyetli projeler gerçekleştirilebilir.Ayrıca basit çözümlerle oluşturulan programın bakımı ve geliştirilmesi de kolaydır. Basit çözümler karşıya kolay aktarılır ve adapte edilir. Bu zaman kazanılması anlamına gelmektedir.  
XP projelerinde kalite kontrolü geridönüm (feedback) üzerinden sağlanır.En önemli özelliklerinden birisidir.Programcılar yazdıkları testlerden geridönüm alarak kaliteyi sağlarlar.Kısa süreç içerisinde yeni sürümler oluşturarak kullanıcı geridönümlerinden programın yeterliliğini saptar.Bu sebeple canlılık için su ne ise XP modelimiz için de geridönüm odur.  
Bu modelde tüm proje çalışanları birbirleriyle etkileşim ve iletişim halindedir.Bireyler arası yüzyüze görüşmeler önem teşkil eder.  
Bu modelde cesaret yapılan işlerde gerekirse hemen değişikliği göze alabilmeyi simgeler.Bu klasik yöntemler için zordur çünkü yapılan değişiklerin doğurduğu kayıplar ve maaliyetler yüksektir.Fakat XP modeli için göğüslenebilir durumdur. Nihayetinde başarısızlıktan korkmak yerine başarısızlığı oluşturan nedenler üzerine gitme cesaretini göstermek doğru olacaktır. Başarısızlıkla mümkün olunan en kısa sürede karşılaşmak, telafi etme şansını arttırır.  
  
**SCRUM**  
**Scrum**, günümüzde kullanılan en popüler agile frameworklerden birisidir ve kompleks ürünler, sistemler geliştirmek için kullanılır.Büyük resmi parçalara böler ve her bir parçayı geliştirerek bir yol izler.Scrum teorisi deneysel süreç kontrol teorisine dayanmaktadır. Bu teoriye göre, bilgi deneyimle elde edilir ve deneyimlerin ışığında verilen kararlar bizi başarıya götürür.Bu teorinin 3 temel ilkesi:  
**Şeffaflık;** Projenin ilerleyişi, sorunlar,gelişmeler herkes tarafından görülebilir olmalıdır.  
**Denetleme;** Projenin ilerleyişi düzenli olarak kontrol edilir.  
**Uyarlama;** Proje, yapılabilecek değişikliklere uyum sağlayabilmelidir.

Rugbydeki mantığı da tüm takım oyuncularıyla birlikte karşı sahaya taşınarak atak yapması olan scrum adını bu takım oyunundaki hücum tekniğinden almıştır.  
Scrum aşağıdaki 3 temel kavram çerçevesinde hareket etmektedir:  
**1)Roller:**  
Ürün sahibi(Product Owner): Projenin geri dönüşlerinden sorumlu kişidir. Bu kişi müşteri olmak zorunda değilidir. bazı durumlarda müşterinin kendisinin, bazı durumlardaysa müşterinin isteklerini aktaran kişidir. Ürün olarak ne istediğini, üründen ne beklediğini ve amaçlarını yazılım ekibine aktarır.  
Scrum yöneticisi(Scrum Owner):Bu roldeki kişiler scrum kurallarını iyi bilmelidir. Scrum sürecinin yönetilmesinden, takıma koçluk yapmaktan ve takımın motivasyonunu yüksek tutmaktan sorumludur.  
Scrum Team:Ürünü oluşturan kişilerin oluşturduğu ekiptir.Ekibin üyeleri farklı becerilere,uzmanlıklara ve deneyimlere sahip olsalar da ekibin içinde unvanlar yer almaz.  
Özetleyecek olursak, Product owner, yani ürün sahibi bitirmesi gereken işleri tanımlamaktan sorumludur. Scrum master, yani geliştirme ekibi bu işlerin nasıl tamamlayacağından ve tamamlamaktan sorumludur. Scrum master ise tüm sürecin kurallara uygun ve sorunsuz bir şekilde işletilmesinden sorumludur.  
Scrum terimlerine geçecek olursak "**Sprint** nedir?" sorusuyla başlayalım.  
Sprint, belirli bir işin tamamlandığı ve incelemeye hazır hale getirildiği sınırlandırılmış bir zaman dilimidir. Sprintler genellikle 2-4 hafta uzunluğundadır ancak bu projeye ve ekibe bağlı olarak değişmektedir.Scrum, birbirini takip eden sprintlerden oluşmaktadır ve her bir Sprint sonunda müşteriye sunulabilecek bir iş çıkarılmalıdır.  
**Sprint Backlog:**Bir sprintte yapılacak işlerin yer aldığı ve Product Backlog’daki öncelik sırasına göre, tüm ekibin beraber oluşturduğu bir listedir. Kısaca bir sprintte gereksinimler sonucu ortaya çıkan iş listesidir.  
**User story:** yapılacak işin son kullanıcı açısından anlaşılır bir şekilde açıklamasıdır.  
**2)Toplantılar**  
**Sprint Planning (Planlama Toplantısı):** Sprint başlamadan önce Product Owner, Sprint için bir hedef tanımlar ve hangi User Storylerinin tamamlanması gerektiğini belirler. Product Owner ile Development Team birlikte çalışarak, hangi işlerin gelecek Sprint'e alınacağına karar verir. Geliştirme Ekibinin kapasitesini aşması halinde daha küçük parçalara ayrılması gereken işler varsa küçük parçalara ayrılır ve bir sonraki Sprint hazır hale getirilir.  
**Daily Scrum Meeting (Günlük Sprint Toplantıları):**Scrum takımı olarak günlük yapılan kısa toplantılardır.Bu toplantıda o gün içerisindeki yapılacak işler ve problemler hakkında konuşulur.  
**Sprint Review (Demo Toplantısı):**Toplantıda geliştirme ekibi herkese bir Sprint boyunca neler yaptığını ve ortaya çıkan ürünü gösterir. Her Sprint sonrası gerçekleşen Sprint Review'de Development Team, Scrum Master ve Product Owner bulunur.Buradaki amaç ürünün gereksinimlere uygununluğunun gerçekliğinden emin olmaktır.  
  
**3)** **Bileşenler/Araçlar**  
**Ürün Gereksinim Listesi:**Proje boyunca yapılacak olan iş elemanlarının listesidir.Gelişebilen ve değişebilen bir dökümandır.  
**Sprint Kalan Zaman Grafiği (Burndown Chart):** Sprint süresince ne kadar iş yapıldığını ve ne kadar yapılması gerektiğini karşılaştıran grafiktir.  
  
Scrum yöntemi günümüzde büyük şirketler olan Google, Microsoft, BBC vb. başta olmak üzere çoğu şirkette kullanılmaktadır.

Kaynak:  
<https://medium.com/@denizkilinc/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-temel-a%C5%9Famalar%C4%B1-software-development-life-cycle-core-processes-197a4b503696>

<https://medium.com/@secilcor/scrum-nedi%CC%87r-6a4326951dd8>

<http://ybsansiklopedi.com/wp-content/uploads/2015/08/Yaz%C4%B1l%C4%B1m-Geli%C5%9Ftirme-Modelleri-Yaz%C4%B1l%C4%B1m-Ya%C5%9Fam-D%C3%B6ng%C3%BCs%C3%BCSDLCYBS.pdf>[https://www.yazilimtestmerkezi.com/post/yazilim-gelistirme-modeli-nedir-cesitleri-nelerdir](https://www.yazilimtestmerkezi.com/post/yazilim-gelistirme-modeli-nedir-cesitleri-nelerdi) MELİSA ERDEM 210601010