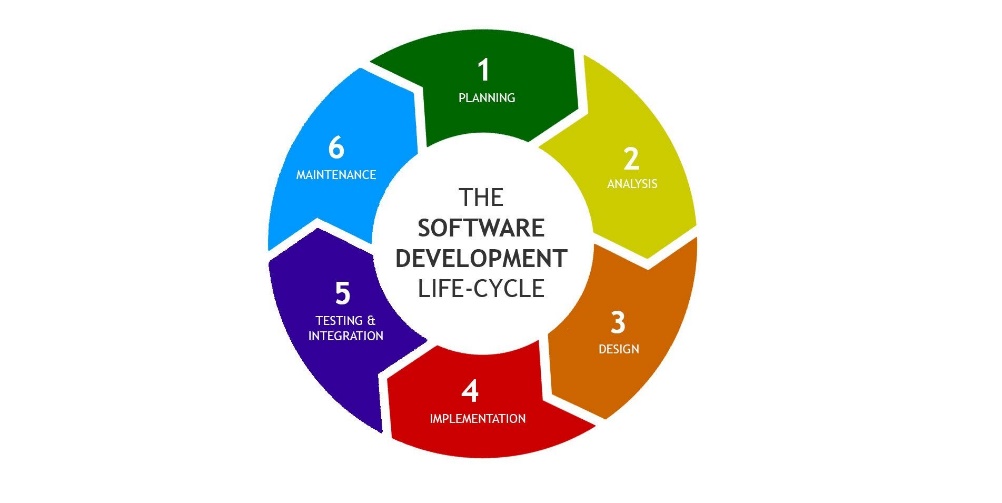
Yazılım Yaşam Döngüsü (SDLC)

Ve Modelleri

Alp ALTAN 210601068

**Yazılım Yaşam Döngüsü (SDLC) nedir?**

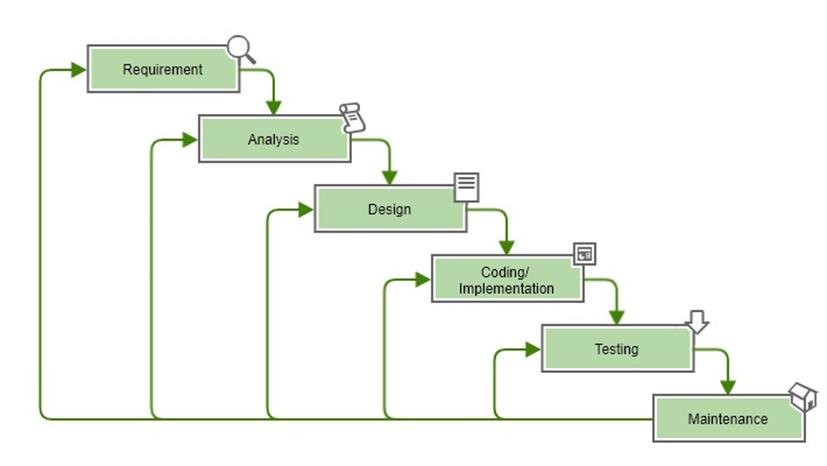
Yazılımın sözlük anlamına baktığımızda “Bir bilgisayarı belirli işlevleri yerine getirmek üzere yöneten, bilgisayara ne yapacağını söyleyen, kodlanmış komutlar dizisi.” ifadesiyle karşılaşırız. Yazılım kodlanmış komutlar dizisinden ibaret değildir. Günümüz Dünyasında teknoloji oldukça hızlı ilerlemektedir tüketim ve istekler her gün artar bu duruma sadece insanlar ayak uydurmakla kalmaz teknolojinin de ayak uydurması gerekir. Gelişen Dünyada farkında olsak da olmasak da işimizi kolaylaştıran sayısız yazılım var. Artık insanlar ellerini ceplerine atarak bile milyarlarca satır koddan oluşan yazılımlara erişebiliyorlar peki hiç düşündünüz mü, siz farkında olmasanız bile sürekli güncellenen, değişen ve gelişen yazılımlar bu günlere nasıl geldi?

Satırlarca kod yazılım olmadan önce birçok aşamadan geçiyor; Gereksinim, Analiz, Tasarım, Gerçekleştirme, Bakım. İlk olarak gereksinim aşamasında yazılım geliştiricilere istenilenler belirtilir karşılıklı fikir alışverişler ile gereksinimler belirlenir. Yazılımı gerçekleştirecek olanlar bu gereksinimler üstünden fizibilite çalışmasına başlar. Analiz aşamasında veriler toplanır, tasarım hakkında fikirler alınır, projenin yürürlüğü için gerekli olan her şey ayrıntılı şekilde incelenir. Tasarım aşamasına gelindiğinde projenin nasıl ilerleyeceği netleşmeye başlamıştır bu sebeple belirlenen gereksinimlere yanıt verebilecek temel yapı oluşturulur. Gerçekleştirme aşamasında projenin kodlamasına ve birleştirilmesine başlanır. Testler yapılır hatalar kontrol edilir ve proje teslim edilir. Bakım aşaması projenin tesliminden sonra hata giderme ve değişen gereksinimleri projeye uygulama aşamasıdır. Teorik olarak bir yazılım projesinin başından sonuna kadar bütün aşamalar bunlardır fakat o yazılımı gerçekleştirenler ve talep edenler insan oldukları, insanlığın doğasında dinamiklik de olduğu için bu şekilde gerçekleşip bitmez.

Müşteri, gereksinimlerini proje esnasında veya sonrasında değiştirebilir, projenin gerçekleştirim ve analiz esnasında hatalar yapılabilir. Yazılımın bu durumlara uyum sağlaması gerekir. Kısaca Yazılım yaşam döngüsünü üretim ve kullanım aşaması birlikte olmak üzere geçirdiği tüm aşamalar olarak ele alabiliriz.

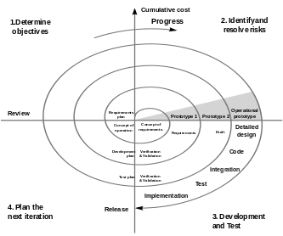
**Yazılım Yaşam Döngü Modelleri nelerdir?**

Yazılım gerçekleştirmede kullanılan birçok farklı yöntem vardır; Şelale (waterfall) modeli, V model, Gelişigüzel model, Barok modeli, Spiral model, Scrum vb. Bu modellerin kullanımı şirketten şirkete, insandan insana değişebilir. Her modelin avantajları ve dezavantajları var. Şirketler projelerine ya da alışık oldukları çalışma biçimine göre farklı modellerle proje yürütebilirler. Günümüzde popüler olarak Scrum modeli kullanılmakta. Birkaç modeli açıklayacak olursak:

Şelale modeli geçmişte en popüler olan model olmakla birlikte günümüzde kullanımı gittikçe azalmaktadır. Şelale modelinde her aşama sonunda test ve dokümantasyon yapılmaktadır, eğer ikisinden biri yapılmazsa o aşamanın tamamlandığı kabul edilmez. Bu modelde aşamalar arasında geri dönüşlerin nasıl olacağı tanımlıdır ancak analiz aşamasında kullanıcının gereksinimleri ayrıntılı bir şekilde belirlenmezse tasarım aşamasında temeli kurarken eksikler oluşur. Bu sebeple en fazla zaman bu iki aşamaya ayrılır. Bir projede somut olarak ilerleme ne kadar geç olursa çalışanlar mutsuzlaşmakta ve işin ilerleyiş hızı azalmaktadır. Kullanıcı bu modelde sürecin içerisinde yer almadığından proje sonunda geri dönüşleri ve maliyeti arttırır.

V model projeyi başından sonuna iki (üretim ve sınama) aşamaya böler bu aşamaların kendi içerisinde ortak üç aşaması vardır; Kullanıcı modeli, Mimari model, Gerçekleştirim modeli. Kullanıcı modeli projenin gereksinimlerini ve sınama aşaması için gerekli verilerin alındığı aşamadır. Mimari modelde ise sistemin tasarımı belirlenir, sınama aşamasında istenilen veriler toplanır. Gerçekleştirim modelinde ise yazılım modüllerinin kodlanması ve projenin sınama aşaması için son haline getirildiği aşamadır. V modeli belirsizliklerin ve proje sonu hataların az olduğu bir modeldir. BT projeleri için uygun bir modeldir.

Gelişigüzel modeli bir model olarak adlandırmak çok doğru olmaz. Bir ekip ya da şirket projeleri için uygun ve tercih edilen bir model değildir. Tek kişilik projelerde geliştiricinin isteğine göre gittiği aşamalardan oluşur. Proje sonrasında geliştirme ve bakım çok zordur.

Barok modeli 1970’li yıllarda ortaya çıkan ve doğrusal bir şekilde ilerleyen bir modeldir. Bütün süreçte aşamalar arası geri dönüşlerin nasıl yapılacağı tanımlı değildir. Bu modelde dokümantasyon ayrı bir süreç olarak ele alınır ve proje sonunda yapılır. Günümüz projelerinde kullanılan bir model değildir.

Spiral model, bütün bir projeyi kendini tekrarlayan dört aşamaya böler; Planlama, Risk analizi, Üretim, Kullanıcı değerlendirme. Kullanıcı bu modelde bütün sürecin içerisinde olduğu için gereksinimlerin proje sonrasında değişmesi ihtimalini azaltır. Bu modelde analiz tasarım gereksinim diye doğrudan bir aşama yoktur. Kendini yineleyerek arttıran bir modeldir bu sebeple kullanıcıya her döngüde prototip teslim ederek ona göre bir yol izlenir.

Artımsal geliştirme modelinde bütün bir projeyi tek seferde teslim etmek yerine parçalara bölerek teslim etmek esas alınır. Kullanıcının istediği gereksinimler önceliklendirilerek çalışmalar düzenlenir. Her parça önceden belirlenen gereksinimler doğrultusunda geliştirilir ve bitip teslim edilene kadar gereksinimlerde değişiklik olmaz. Her parça sonrası kullanıcı değerlendirmesi ile gereksinimler düzenlenir ve her parça bir önceki parçayı kapsayacak şekilde geliştirilir. Bu modelde bir taraftan kullanım yapılırken diğer taraftan üretim yapılabilir.

**Yazılım Yaşam Döngü Modellerinin Karşılaştırılması**

Her yazılım yaşam döngü modelinin kendine ait avantajları ve dezavantajları vardır. Örneğin A projesine Şelale modeli daha uygundur projeyi daha hızlı ve avantajlı şekilde bitirmemizi sağlar fakat proje sonunda değişen gereksinimler ve hata çıkma ihtimali daha fazladır. Bu projeyi Şelale modeli ile değil de Spiral model ile yaparsak projenin gerçekleştirme süresi artabilir ama belirlenen teslim süresi, bütçeyi aşma ve proje sonrası gereksinim değişimi risklerinin ihtimali azalır.

Şelale modelinin avantajları; Kullanımı ve anlaması basittir, yönetimi kolaydır, küçük ve iyi analiz yapılmış projelerde oldukça iyidir. Dezavantajları; Büyük ve nesneye dayanan projeler için kullanımı uygun değildir, gelişim ve değişime çok açık değildir bu yüzden ilk teslimde yüksek memnuniyet oranı düşüktür.

V modeli avantajları; Proje yönetimi takibi kolaydır, hata oranı düşüktür. Dezavantajları; Her aşama bir kere yapılır tekrar edilmez, Projeyi bölüp aynı anda birçok tarafını geliştirme imkanı yoktur, Risk çözümleme yoktur, Geliştirme esnasında gereksinimler değişebilir bu durumda proje tekrar gözden geçirilmelidir zaman kaybına yol açar.

Spiral model avantajları; Kullanıcı erkenden sistemle alakalı bilgi sahibi olur, geliştirme küçük parçalara bölünür, pek çok yazılı modeli için uygundur, hatalar erkenden giderilir bu sebeple teslim sonrası hata geri dönüşleri az olur, riske duyarlı bir şekilde geliştirilir. Dezavantajları; Karmaşık bir süreci vardır, sonu olmayan bir döngüdür, ara adımları çok olduğundan dolayı dokümantasyon çoktur, yazılımın üretim süreci uzundur, pahalı bir modeldir.

Artımsal geliştirme modeli avantajları; Gereksinimler müşteri ile birebir belirlenir, aynı anda hem kullanım hem üretim yapılabilir, kullanıcı sürecin içindedir teslim sonrası geri dönüş ihtimalini azaltır, ilk teslimden itibaren kullanılabilir. Dezavantajları; Bütün sistemin net bir şekilde bilinmesi gereklidir, deneyim gerektiren bir modeldir, artımlar kendi aralarında tekrarlanamaz.

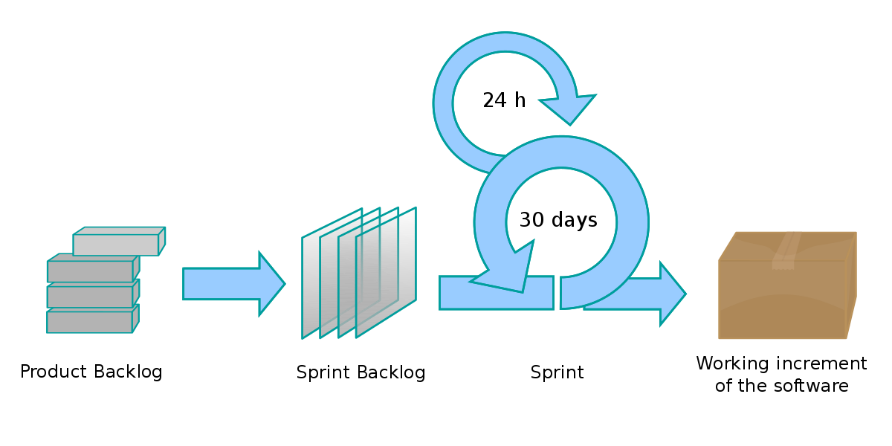
**Çevik (Agile) Yazılım Geliştirme**

Çevik yazılım geliştirme yönetiminde proje ne kadar büyük ya da küçük olduğu fark etmeksizin proje küçük yinelemelere ayrılır ve her yineleme yeni bir projeymiş gibi ele alınarak öyle geliştirilir. Çevik ile here bir yinelemenin 2-4 hafta sürmesi planlanmaktadır. Çevik tekniğinin hızı projenin ekibinde olan kişilerin birbiri ile sürekli iletişim halinde olmasına bağlıdır.

Çevik yazılım geliştirme manifestosu 2001 yılında Kent Beck ve arkadaşları tarafından oluşturulmuştur. Süreçler ve Araçlar yerine Bireyler ve Etkileşimler, Kapsamlı belgeler yerine Çalışan yazılım, Sözleşme görüşmeleri yerine Müşteri ilişkileri, Plan izleme yerine Değişikliğe açıklığın daha önemli ve öncelikli olduğunu belirtmektedir.

Çevik yazılım geliştirmede bazı temel prensipler vardır. Müşteri her zaman önceliklidir ve ilk yapılması gereken müşteriye hızlı ve sürekli olarak teslimat yapabilmektir. Projenin kodlarının yazılmaya başlamasından itibaren gereksinim değişiklerine açık ve uygun esneklikte olmalıdır. Kısa zamanlarda teslimat yapmak önemlidir. Yazılımı geliştirenler arasındaki iletişim kaliteli bilgi akışı için kesilmemelidir. Çevik yazılım geliştirmek için birlikte iyi çalışabilen, mutlu ekip olması gereklidir. Aksi halde aradaki iletişim zorlaşır, projenin gelişmesi yavaşlar ve Çevik olmaktan çıkar. Çevik yazılım geliştirmenin avantajları hem müşteri tarafı için hem de yazılımı geliştiren ekip için çoktur. Ekip içerisinde üretkenlik artar, ekibin motivasyonu yüksek olarak devam eder, üretilen ürünün kalitesi artar, maliyetleri düşer ve kısa sürede müşteri memnuniyeti sağlanır. Çevik yazılım geliştirmede kullanılan birçok yaygın model vardır. Extreme Programming (XP), Scrum, Agile Unified Proccess vb.

**Scrum nedir?**

Scrum aslında Rugby sporunda bir hücum taktiğidir. Top bu taktikte tüm oyuncularla birlikte karşı sahaya taşınmaktadır. Yazılım dünyasında Scrum oldukça popüler bir metottur. Scrum 1990’ların ortalarında geliştirilmeye başlanmıştır. Karmaşık yazılım sistemlerini sprint adı verilen küçük parçalara bölerek geliştirir.

Product Backlog: Müşteriyle birlikte belirlenen ve öncelikli olarak yapılması istenilen gereksinim listesidir.

Sprint Backlog: Yaklaşık 30 günlük zaman dilimlerinden oluşan zaman dilimidir genelde her bir proje 1’den fazla sprintten oluşur.

Scrum önceden müşteriden alınan gereksinimleri sprintlere bölerek 2-4 hafta arasında tamamlamayı hedefleyen ve her gün projenin gidişatı ve özeti ile ilgili toplantıların yapıldığı bir metottur. Scrum’da üç temel kavram vardır: Roller (Roles), Toplantılar (Meetings), Bileşenler/Araçlar (Artifacts).

Roller kendi içerisinde ürün sahibi, scrum yöneticisi ve scrum takımı olarak üçe ayrılır. Projenin scrum takımı ve şirket yöneticileri arasındaki iletişimle ilgilenir. Scrum yöneticisi takımdan bir üyenin Scrum’ın temel bağlarına bağlı kalarak ilerleyip ilerlemediklerinden sorumludur. Scrum takımı işi yapan ekip olarak da adlandırılabilir.

Toplantılar Sprint planlama, gözden geçirme ve günlük Scrum toplantıları olarak üçe ayrılır. Planlama aşamasında geniş kapsamlı gereksinim listesi çıkarılır, başarılı geliştirme için risk değerlendirmesi ve kontrolleri belirlenir. Proje gözden geçirilir ve olası değişiklikler yapılır. Genel bir maliyet hesabı yapılır. Gözden geçirme aşamasında her sprint öncesinde genel bir toplantı yapılır ve sprint boyunca önceliklerin belirlenmesi sağlanır. Toplantılar her gün ve gün içinde yapılanlar ya da yapılacaklar önceden yapılmışlar kalanlar gibi bilgilerin paylaşıldığı kısa toplantılar.

Özetle Scrum Dünyada anlık olarak en çok tercih edilen Çevik metodudur. Kullanıcıya belli aralıklarla bilgi vermesi, Scrum ekibinin sürekli iletişimde olup ileriyi planlaması, ekibin motive kalıp herkesin birleşip tek bir iş uğruna uğraşması bu metodu bu kadar başarılı kılmıştır. Sonuçta o yazılımı isteyenler de yapanlar da insan ve insan istediği ve gerekli ortamlar sağlandığı sürece yapamayacağı şey yok.

**Scrum günümüzde neden bu kadar popüler?**

Birçok büyük firma (Microsoft, Apple, Google, IBM, Nokia vb.) halihazırda scrum metodu ile çalışıyor. Bunun sebeplerinden biri de yüksek başarı oranı olmasıdır. Bir şirket ne kadar büyükse kazançları ve kayıpları da o kadar büyük olur. Büyük firmalar risk almayı pek sevmezler bu sebeple bir projeye ya da işe başlarken uzun bir süre araştırma ve analiz yapılır. Scrum yüksek başarı oranı, kolay takip edilmesi, kolay düzenlenmesi ve geliştirilmesi yönü ile büyük firmaların dikkatini çeken ilk yazılım geliştirme modellerinden/metotlarından biridir. Büyük firmalar bir şeyler başardıkça diğer büyük ya da küçük fark etmez diğer firmalar bu durumu görüp örnek alırlar. Bu şekilde scrum daha da yayılmaya ve başarılı olmaya devam etmiştir.

İletişim:

* [alpaltnceng@gmail.com](mailto:alpaltnceng@gmail.com)
* [210601068@bakircay.edu.tr](mailto:210601068@bakircay.edu.tr)
* <https://medium.com/@alpaltnceng>
* <https://github.com/alpaltan/>