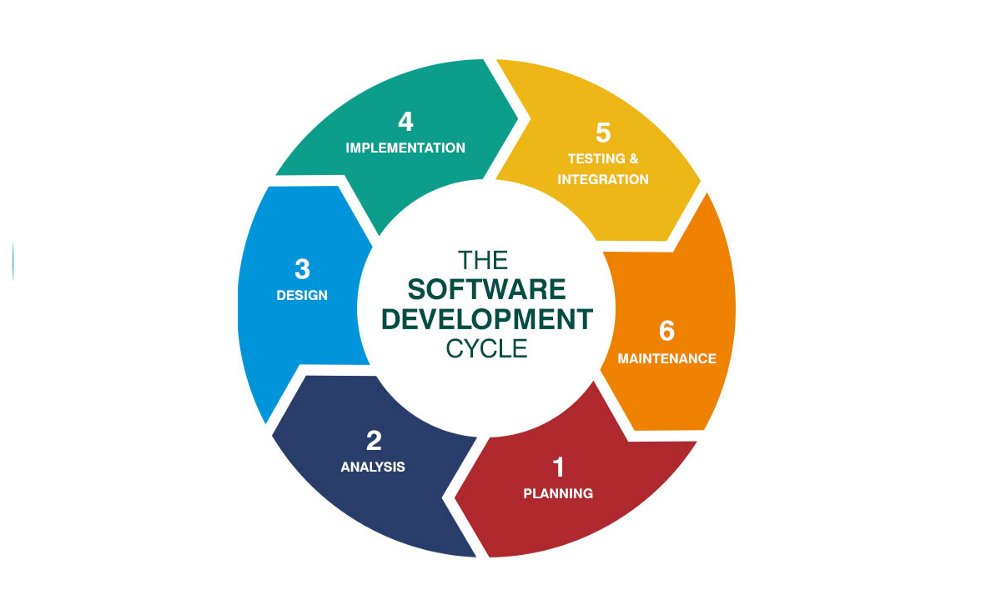
**YAZILIM YAŞAM DÖNGÜ MODELLERİ(Software Development Life Cycle)**

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

Yazılım da bir üründür. Her ürünün üretilmesi için belirli bir plan yapılır ve bu aşamalar izlenmesi gerektiği gibi yazılım projelerinin de sürdürülebilir, en kısa sürede ve kalitesi yüksek projeler ortaya konması için uygulanması gereken birkaç temel aşamadan oluşur. Sistematik bir şekilde ilerlememizi sağlayan tüm aşamalara yazılım yaşam döngüsü(SDLC) olarak adlandırılmaktadır. Yazılımın işlevleri ve ihtiyaçları sürekli değişmesi ve gelişmesi sebebiyle bu aşamalar doğrusal olarak değil bir döngü olarak gerçekleştirilir. Döngü içerisinde herhangi bir evreye geri dönülebilir. Bu döngüler sayesinde yazılımın gelişimi daha rahat ve planlı şekilde gerçekleştirilir. Döngü bazı temel adımlardan oluşur. Planlama, Analiz, Tasarım, Gerçekleştirme, Test ve Bakımdır. Kısaca; bir yazılımın hem üretim, hem de kullanım süreci boyunca süren bir döngüdür diyebiliriz.

**Yazılım Yaşam Döngüsü Adımları:**



1-Planlama(Planning): Gereksinim safhası(requirements) olarak da geçen bu aşamada ilk olarak gereksinimler belirlenir.Developer ile müşterinin iletişim halinde olduğu aşamadır. Üretilecek yazılım ürünü için müşterinin gereksinimleri alınır. Daha sonra ürün için fizibilite çalışması yapılır. Temel gereksinimler doğrultusunda ne istenildiğinin planlanması yapılır. Görev paylaşımın yapıldıgı,fikirlerin öne atıldığı ve tartışıldığı kısımda bu adımdadır. Yapılan planlar doğrultusunda diğer adımlara yön verilir.

2-Analiz(Analysis) : Sistem gereksinimlerin ve yazılım işlevlerini netleştirmek ve bunları dokümante etmektir. Oluşabilecek sorunlar hakkında değerlendirmeler yapılır. Ayrıca bu aşamada ekip çalışması çok önemlidir. Müşteri , yazılım mühendisi, sistem analisti, iş analisti, ürün yöneticisi vb. rollere sahip kişiler bir araya gelerek çalışmayı sürdürür.Kısacası;analiz aşamasında projenin detayları ortaya konulmalıdır.

3-Tasarım(Design) : Analiz aşamasında belirlenen gereksinimler doğrultusunda proje çizilir. Yazılım sistemin temel yapısının oluşturulduğu adımdır. Tasarım aşaması iki bölümden oluşur: mimari tasarım ve ayrıntılı tasarımdır. Mimari tasarımda yazılım ürününün genel bir planı yapılır ve modüller belirlenir. UML diyagramları ve iş akış şemaları bu adımda çizilir. Ayrıntılı tasarımda ise yazılımda kullanılacak algoritmalar, programlama dilleri , veri tabanları ve bunun gibi detaylar belirlenir.

4-Gerçekleştirim(Kodlama ve Test) : Müşteriye teslim edilecek olan yazılım ürünün programlama aşamasıdır. Yazılan kodlar ‘Clean Code’ felsefesine uygun bir şekilde yazılmalı. Yani amaç her zaman okunabilirliği yüksek, bakımı kolay ve esnek yapılar oluşturmaktır. Kodlama yaparken ve sonrasında yapılan önemli bir adımda test etmektir. Yazılım ürününün erken test edilmesi zaman ,maliyet ve prestij gibi kayıplar yaşamamızı sağlar. Daha sonra da ürün piyasaya veya müşteriye sunulur.

5- Teslim ve Bakım: Tüm aşamaların tamamlanmasının ardından yazılım ürünü piyasaya veya müşteriye sunulur. Ürün teslim edilirken ürünle birlikte kullanım kılavuzu da müşteriye teslim edilir. Ürünün teslim edilmesi ile birlikte bakım aşaması başlar ve yazılım ürününün ömrü süresince devam eder. Bu süreçte hataları giderme, ürüne yeni özellikler ekleme, altyapıları iyileştirme gibi bakım işlemleri yapılır.

**Yazılım Yaşam Döngü Modelleri:**

Yazılım yaşam döngüsünde ürün ihtiyacına göre çeşitli modeller kulllanılır.Bu modeller yazılım geliştirmeyi sistematik hale getimek,düzenlemeyi kolaylaştırmak amacıyla ortaya çıkmıştır. Modeller o dönemin donanım ve sektörün ihtiyaçları doğrultusunda ortaya çıkmıştır.

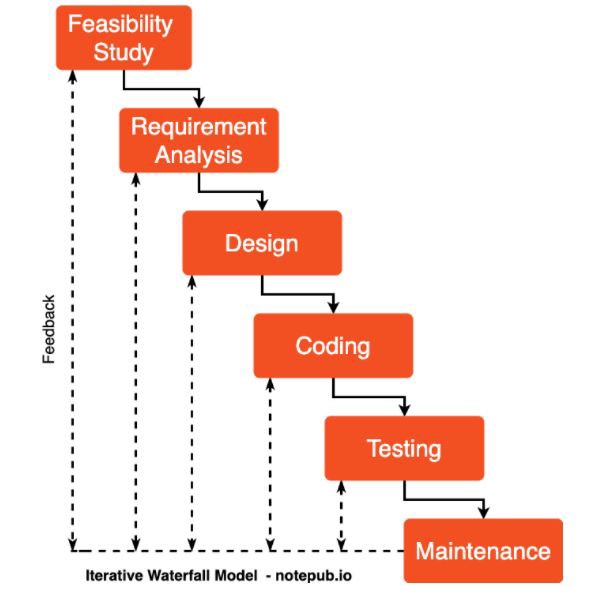
**1-Gelişigüzel Model :**

1960'lı yıllarda kullanılan bu model tam bir model sayılmaz. Bu modelde belirlenmiş bir model ya da yöntem bulunmaz. Bu sebeple gelişigüzel geliştirme sürecinde yazılım geliştirmek daha zor ve karmaşıktır. Genellikle basit programlama içeren ve tek bir kişi tarafından geliştirilen yazılım ürününde kullanılan yöntemdir.

**2-Barok Modeli :**

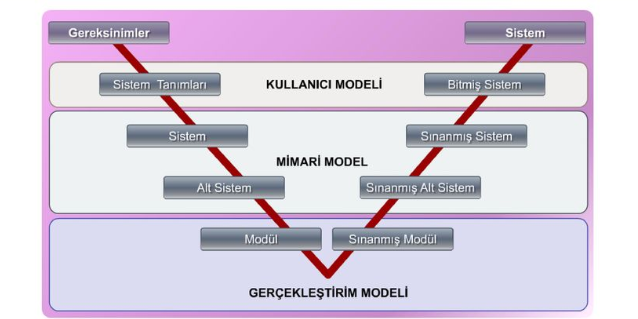
1970'li yıllarda ortaya çıkan bu modelde yazılım yaşam döngüsü aşamaları doğrusal bir şekilde uygulanır ve aşamalar arası nasıl geri dönüş yapılacağı belirsizdir. Bu modelde diğer modellerden farklı olarak belgelemeyi ayrı bir süreç olarak ele alır. Yazılımın geliştirme ve test faaliyetleri tamamladıktan sonra yapılır. Gerçekleştirim aşamasına daha fazla ağırlık veren bir model olup, günümüzde kullanımı önerilmemektedir.

**3-Çaglayan Yaşam Döngü Modeli :**



Şelale(waterfall) modeli olarak da bilinen bu model en eski, en tanınmış ve en temel modeldir. Geleneksel modelde denmektedir.Bu modelde aşamalar en az bir kez tekrar edilmedilidir.Kullanımı ve anlaması basit, yönetimi kolaydır. Küçük ve gereksinimi iyi tanımlanmış projelerde etkili çalışır. Barok modelden farklı olarak belgeleme işi üretimin doğal bir parçasıdır ve barok modelin tersine burada aşamalar arası geri dönüşlerin nasıl yapılacağı tanımlıdır. Bu modelde aşama aşama ilerlenir ve bir aşama bitmeden sonraki aşamaya geçilemez. Her aşamada dokümantasyon ve test yapılması gerekir. Eğer dokümantasyon ve test yapılmamışsa o aşama bitmiş sayılmaz. Projelerde gereksinimlerin değişebilmesi olağan bir durumdur. Kodlama ve test aşamalarında yapılacak değişiklikler maliyeti arttırır. Yazılımın oluşması süreci bu modelde oldukça uzun olduğundan yazılımcılar bir an önce programı kodlama ve programı çalıştırma amacında oldukları için ve bu modelde kodlama sadece küçük bir kısımda kullanıldığından ekip mutsuzlaşacaktır. Gereksinimleri belirli ve kısa sürede bitebilecek projelerde kullanılması daha uygundur. Büyük projelerde istekler kesin ve sabit olmadıgından,tek önlü ilerlediği için değişime ve gelişime açık bir model degildir.Dolayısıyla bu projeler için uygun degildir.Günümüzde ise kullanımı gittikçe azalmaktadır.

4- V Süreç Modeli :



V şeklinde olan bu model kodlama aşamasından sonra yukarı doğru eğim aldıgı için oluşmuştur. Çağlayan modelinin biraz daha gelişmişidir. Sol tarafta üretim, sağ tarafta ise test(sınama) işlemleri yer alır. Üretim aşamalarının karşılarında bulunan sınama aşamaları sayesinde hata kaynaklarına dönüş daha kolay olur. Modelin üç temel çıktısı vardır. Yukarıdan aşağıya doğru bakacak olursak;

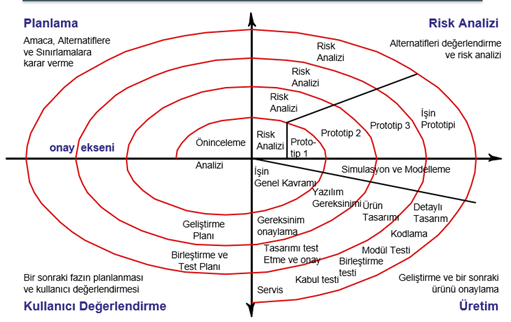
Kullanıcı Modeli: Geliştirme sürecinin kullanıcı ile olan ilişkileri tanımlanır. Sistemin nasıl kabul edileceğine ilişkin kullanıcı kabul testi gibi sınama belirtimleri yapılır. Yazılım ürünün planlaması yapılır. Müşteri ile sürekli iletişim içinde olunur.

Mimari Model: Bu aşamada mimari(tasarımsal) ögeler anlaşılır ve tasarlanır. Tüm sistemin sınaması sağlanır.

Gerçekleştirim Modeli: Yazılım ürünü kodlanır ve denenmesinden oluşur.

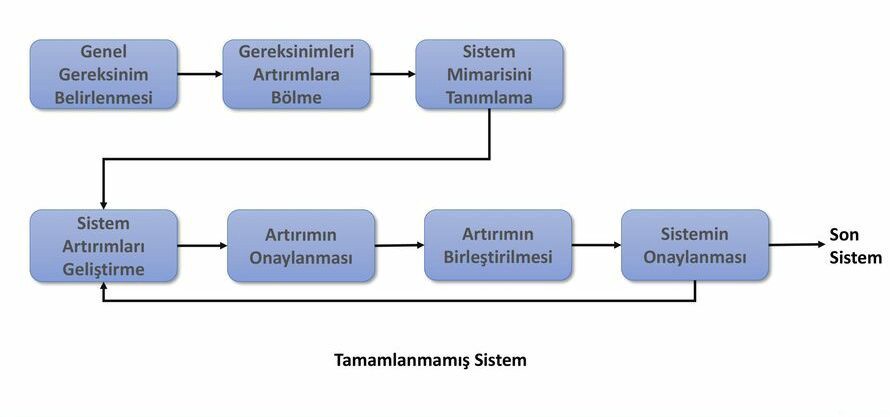
Bu modelde belirsizliklerin az ve iş tanımlamalarının net olduğu Bilgi Teknolojileri için uygun modeldir. Aynı zamanda kullanıcının projeye katkısını arttırmakta, proje takibini kolaylaştırmaktadır. Modelin kullanımı basittir. Aşamalar arası tekrarlamaları kullanmaz ve risk çözümleme gibi aktiviteleri içermez.

**5-Helezonik Model(Spiral):**



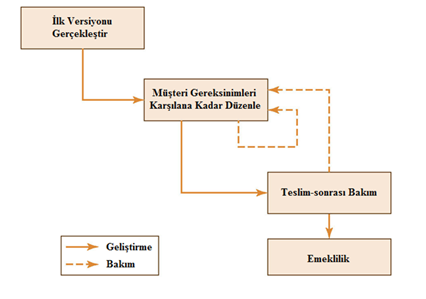
Spiral model de denmektedir. Bu modelde planlama, risk analizi, üretim ve kullanıcı değerlendirmesi olarak 4 ana aşamadan oluşur ve bu aşamalar bir spiral oluşturacak şekilde dönerek tekrar eder. Her döngü bir fazı temsil eder. Çağlayan modelinde göze alınmayan riskleri göz önünde bulundurur. Helezonik modelde kullanıcının sürecin içerisinde olması kullanıcı ara ürün değerlendirmesi ve test aşamalarında aktiftir dolayısıyla sistemi erken tanır. Bu sayede kullanıcının istemediği bir sonucun sonradan fark edilip büyük zararlar oluşması riskini azaltır. Büyük projelerde bu model kullanıma uygundur. Küçük ve düşük riskli projeler için maliyetlidir. Çünkü devamlı olarak prototip çıkarmak ve dokümantasyon yapmak gereklidir.

6-Arttımsal Geliştirme Süreç :



Bu modelde sistem bir bütün değil küçük parçalar halinde ele alınır ve parçalar kullanıcı önceliğine göre sıralanır. Parçalar sıralandığında birer ara ürün geliştirilmiştir ve ürünleri kullanıcı kullanır. Üretilen her yazılım sürümü birbirini kapsayacak ve giderek artan sayıda işlev içerek şekilde geliştirilir. Açıkçası; bu modelde bir taraftan kullanım diğer taraftan üretim yapılır. Uzun zaman sayılabilecek ve sistemin eksik işlevlikle çalışabileceği türdeki projeler için uygun olup, ürün gereksinimlerinin müşterilerle belirlenmesi, erken artımların prototip gibi davranması da gereksinimlerin daha iyi anlaşılmasını sağlar, başarısızlık riskinin az olması da bu modelin avantajları arasındadır. Dezavantajları ise maliyet olarak şelale modelinden yüksektir. Bu modelde deneyimli personel ihtiyacı duyulur. Artırımların tanımlaması için tüm sistemin detaylı bir şekilde tanımlaması gerekmektedir. Artırımların kendi içlerinde tekrar edilmesine izin verilmez.

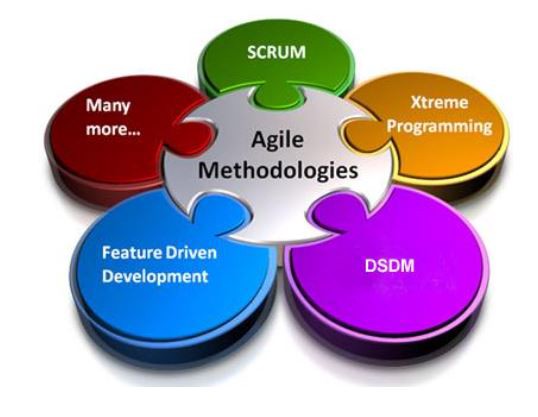
7-Kodla Ve Düzelt Yaşam Döngü Modeli:



Yazılım geliştirmek için en kolay modeldir. İlk aşamada yazılım ürünün ilk sürümü gerçekleştirilir. Daha sonra müşteri gereksinimleri karşılanana kadar düzenlenir. Bu süreç sonunda ürün son halini alır. Sisteme ait dokümantasyon olmadığı için bakım safhası var ama çok zordur. Birkaç yüz satırdan oluşan programlar için kullanılabilir. Ancak dokümantasyona sahip olmadığı için oluşabilecek problemleri çözmek zor ve maliyetli olacaktır. Bu yüzden çok küçük veya tecrübesiz firmalardaki yazılım projelerinde bu model kullanılır.

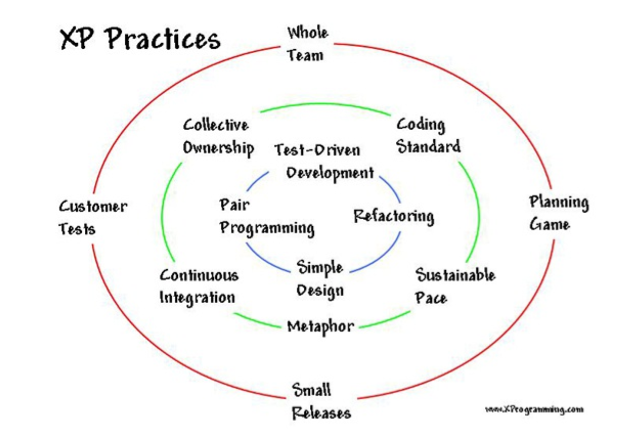
**Çevik(Agıle) Yazılım Geliştirme**

Bilindiği üzere yazılım geliştirme süreci sıkıntılı ve uzun süren bir dönemdir. Yazılım sektöründe, yazılım sürümlerinin zamanında ortaya çıkarılamaması, hataların geç farkedilmesi,degişiklik isteklerine çabuk cevap verilememesi gibi sorunlar ortaya çıkmaktadır. Bu sorunların çözümü için 1990 lı yıllarda çevik(agile) diye adlandırılan metotlar geliştirilmiştir. Bu metotlar sayesinde kısa sürede hata oranı az, yüksek verimli ve ucuz çözümler saglanmaktadır.Bu metodolojide projenin boyutu ne olursa olsun proje küçük yinelemelere ayrılır ve her yineleme birer proje gibi ele alınarak geliştirilir. Bu sayede projede geriye dönük hataların düzeltilmesi kolaylaştırılmaktadır. Yinelemenin sonunda müşteriye projenin ne kadarının gerçekleştirildiği hakkında bilgi aktarımı gerçekleştirilir. Çevik metotlar birçok metodolojileri içerirler. Günümüzde yaygın kullanılan çevik metodolojiler;Extreme Programming (XP) ve SCRUM’dur.



**ÇEVİK METODOLOJİLER**

**1-Extreme Programming (XP):**



Uç programlama olarak da bilinen XP, gelişen müşteri gereksinimlerine yanıt vermeyi ve kaliteyi arttırmayı amaçlamaktadır. Bu model 4 temel değerden oluşur.Bunlar:iletişim, basitlik, geri bildirim ve cesaret.

**1-İletişim :** XP’nin yapıtaşıdır. Yazılım projelerinde çoğu hatanın nedeni ekip içindeki ve müşteriyle olan iletişim kopuklugudur.XP ise iletişim eksikliğini ortadan kaldırmak isteyen bir yöntemdir.XP’de iletişim yüz yüze olmalı ve bilgi alınması gereken bir noktada kısa süre içinde müşterilerle temasa geçilmelidir. İletişimler hızlı olursa projenin de aksama ihtimali ortadan kalkacaktır.

**2-Basitlik :** Sağlanması zor olan bir konudur. Çünkü detaylandırılması gereken kısımlar atlanarak,başka sorunlara sebebiyet verebilir. Karmaşık çözümler XP’nin mantığına uymaz.Yani;XP basitliği en iyi şekilde sağlamak için günün ihtiyaçları dogrultusunda,basit ve esnek bir sistem gerçekleştirmeye çalışır.

**3-Geri Bildirim :** Projenin başarısı için vazgeçilmez bir ihtiyaçtır. Aynı zamansa müşteri proje grubunun bir üyesidir. Belirli aralıklarla (2–4 hafta) müşteri kendisine sunulan sistemin durumunu değerlendirir. Böylece ilerde oluşabilecek sorunlar engellenir ve projenin istenilen doğrultuda ilerlediği kesinleştirilir.

**4-Cesaret :** Xp’nin dört temel değerinden en zorudur. Projelerin üzerine yılmadan gidilmesi, geliştirilmesi ve koddan memnun kalınmazsa onları çöpe atıp yeniden yazabilmelidir. Çünkü XP başarısızlıktan korkmayı degil,başarısızlık durumunda onu en kısa zamanda telafi etmeyi önerir. Unutmayın ki başarısızlıktan korkmak yazılımcının yavaş hareket etmesine sebep olur ve proje hızını da düşürecektir.

Yazılım geliştirmede basitliği ve kolaylığı sağlayabilmek için XP 12 farklı pratiği göz önüne alır. Bunlar; Planlama oyunu, ekipte müşteri, önce test, basit tasarım, çiftli programlama, sürekli entegrasyon, kısa aralıklı sürümler, yeniden yapılandırma, ortak kod sahiplenme, metafor, haftada 40 saat ve kodlama standartıdır.

**Scrum:**

Scrum projeyi“sprint” adı verilen küçük birimlere bölerek geliştirilir. Her bir sprinti teker teker geliştirir. Karmaşık ve gereksinimlerin kolayca tanımlanamadığı projelerde tercih edilir. Bu yöntemde bir yinelemenin tamamlanması 30 günden fazla sürmemelidir. Genel olarak scrum günlük gerçekleşen on beş dakikalık toplantılarla iş takibi yapılır.Scrum genellikle gereksinimlerin kolayca tanımlanamadığı büyük ve karmaşık projelerde kullanılır. Google, Microsoft, IBM, Yahoo, Ikea, Nokia vb. büyük şirketlerde kullanılmaktadır.



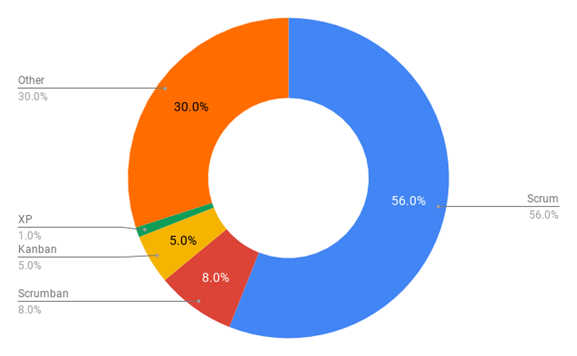
Scrum’da üç temel kavram vardır. Bunlar: Roller, toplantılar ve bileşenler/araçlardır.

**Roller :** Scrum yöneticisi,scrum takımı ve ürün sahibinden oluşur.Ürün sahibi yöneten kişidir.Scrum takımı (scrum team) aynı hedef için çalışan 5–9 kişiden oluşan gruptur.Scrum yöneticisi ise bu takımları scrum çalışmasına göre organize eder.

**Toplantılar :**Scrum takım üyelerinin her gün bir araya gelerek ayak üstü gerçekleştirildiği 15 dakikalık toplantılardır. Bu toplantılarda dün ne yapıldı, bugün kim ne yapacak, karşılaşılan sorunlar nelerdir ve bu sorunlar için nasıl bir yol izlenmeli gibi sorularla toplantı yapılır.

**Bileşenler/Araçlar :** Ürün Gereksinim Dokümanı oluşturulur. Proje süresince yapılması gereken işler listelenir.Sprint Dokümanı oluşturulur. Burada sprint boyunca yapılacak işler takım tarafından detaylandırılarak planı oluşturulur.Sprint dokümanını sadece takımdakiler degiştirebilir.Sprint Kalan Zaman Grafiği ise yapılan işin ne seviyede olduğu ve aslında planlanan zaman aralığına göre nerede olduğunu belirlemek için hazırlanır.

**Scrum Günümüzde Neden Popüler ?**



SCRUM günümüzde en popüler yazılım geliştirme yöntemidir. Sadece yazılım geliştirmede değil birçok sistemin geliştirilmesinde de kullanılmaktadır.Scrum,maliyeti yüksek ve uzmanlık gerektirir. Fakat basit kısa sürmesi, kolay uygulanması ve verimli olması sebebiyle günümüzde kullanımı oldukça artmaktadır. Google, Microsoft, IBM ve Yahoo gibi büyük şirketler tarafından tercih edilmektedir. Bunun yanında Scrum’ un sunduğu takım içerisindeki iletişim, dokümantasyona verilen önem, müşteriyi süreç içerisine dahil etmeside projenin başarı oranını arttırarak popülerliğini de arttırmaktadır.

**Modellerin Karşılaştırılması**

Projede ihtiyaçlar doğrultusunda en uygun model seçilmelidir. Projenin yapısına uygun model seçilmediği zaman projenin maliyeti,zaman,verimliligi fazlaca artar ve projenin başarısız olma ihtimalini arttıracaktır. Bu yüzden projeye uygun model seçilmelidir.

Gelişigüzel ve Barok modelleri dokümantasyon barındırmamaları ve yinelemeli olmamaları sebebiyle günümüzde ihtiyaçları tam karşılayamadığı için kullanımını yitirmiştir.

Çağlayan modeli, eski dönemlerde kullanımı basit, detaylı planlaması ve anlaşılır olması sebebiyle kullanımı yaygındı. Ancak müşteri ile sağlıklı iletişim olmaması sebebiyle, büyük projelerde kullanımı uygun olmayıp, değişikliğin zor olmasından dolayı günümüzde kullanımını yitirmiştir.

V modeli şelale modelinin gelişmişidir. Kullanım kolaylığı, tanımlamaların tam olması gibi avantajları olsada risk çözümleme adımları bulundurmaması gibi dezavantajlara sahiptir.

Helezonik modelde yazılım sürecine müşteri dahil edilir, prototipleri müşteriler değerlendirdiği için hata olasılığını düşürür. Ancak çok zaman alması, dokümantasyonun çok fazla tutulması sebebiyle büyük projelerde kullanılır.

Artımsal geliştirme modelinde gereksinimler müşteri ile belirlenerek, üretilecek yazılım ürünü birbirini kapsayacak ve giderek artan işlev içerecek şekilde geliştirilir. Bu artırımlar kendi içlerinde tekrarlanmadıklarından dolayı değişiklik yapmak zordur. Bu durum modelin kullanımını azaltmaktadır.

Kodla ve düzelt modeli küçük projelerde kullanılır. Proje tesliminden sonraki bakım safhasını yapmak çok zor ve maliyetlidir. Genelde tek bir kişinin kendi metotlarıyla, kendi istediği gibi yapılacak projelerde kullanılır, dokümantasyon gibi olaylar yoktur.

Çevik modellerde takım içi iletişim, müşteri ile iletişim, zaman kullanımı çok önemlidir. İletişime önem verildiği için hata oranı azalır. Değişime açık ve esnektir ancak dokümantasyon detaylı olmadığı için alışılmışın dışındadır. Günümüzde en çok kullanılan bir modeldir.

**Kaynakça:**

*Doç. Dr. Deniz Kılınç, İzmir Bakırçay Üniversitesi Yazılım Mühendisliği Temelleri 2. ve 3. Hafta Sunumları*

<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://www.codex.com.tr/yazilim-gelistirme-modelleri>

<https://bilgisayarkavramlari.com/2014/05/12/sdlc/?highlight=sdlc>

<https://www.linkedin.com/pulse/sdlc-yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-nedir-abdulsamed-bah%C3%A7aci/?originalSubdomain=tr>

<https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/>

<https://www.fatihyilmaz.com.tr/yazilim-gelistirme-yasam-donguleri-system-development-life-cycle-sdlc-nedir/>

<https://zeynepaygun.wordpress.com/2017/05/29/what-is-sdlc-sdlc-nedir/>

Yu Beng Leau, Wooi Khong Loo, Wai Yip Tham, Soo Fun Tan, "Software Development Life Cycle AGILE vs Traditional Approaches", Int. Conference on Information and Network Technology, IPCSIT Singapore, Vol. 37, pp. 162, 2012.

Doç. Dr. Recep ERYİĞİT , “*Yazılım Mühendisliği Süreç Modelleri*” , Ankara Üniversitesi

Doç. Dr. Resul DAŞ, “*Yazılım Yaşam Döngüsü ve Süreç Modelleri*”, Fırat Üniversitesi

Sultan SARIZEYBEK