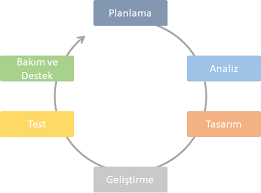
**YAZILIM YAŞAM DÖNGÜSÜ**  Yazılım yaşam döngüsünü üzerinde çalışılan yazılımın geliştirilmesi, test edilmesi ve bakımını kapsayan bir süreç olarak özetleyebiliriz. **YAZILIM YAŞAM DÖNGÜSÜ ADIMLARI**



Şekil 1 : Yazılım Yaşam Döngüsü Adımları

Şekil 1’de de görüldüğü üzere bir yazılım yaşam döngüsünün adımları vardır. Bu adımlar aşağıdaki gibi sıralanabilir.

1)PLANLAMA: Yazılım oluşturma sebebimiz olan problem bulunur. Ona yönelik çözüm önerileri üretilir ve bu öneriler üzerinden proje planlanır, çalışanlar arası görev dağılımı yapılır.

2)ANALİZ: Projenin gereksinimleri, detayları, riskleri, maliyeti, kullanılacak programlama dili vb. konuşulur, tasarlanıp gerçekleştirmek üzere ortak karara bağlanır. Projeyi gerçekleştirmenin kolaylaşması için gereksinimler net olmalıdır.

3)TASARIM: Bu aşamada proje çizimleri yapılır. Kararlar bir sonraki aşama gerçekleştirme olacak şekilde verilir. Tasarımın mimari ve detaylı olmak üzere iki aşaması vardır. Mimari tasarımda projenin yapısı, kullanılacak modeller belirlenirken detaylı tasarımda veri yapıları, arayüzler gibi ayrıntılar belirlenir. UML diyagramları/akış şemaları burada kullanılır.

4)GELİŞTİRME: Kodlama aşaması diyebiliriz. Çeşitli teknolojiler ve programlama dilleri kullanılarak tasarlanan proje kodlanır. Şekil1’ de test aşaması ayrı şekilde gösterilmiş olsa da test aşamasını geliştirme aşamasına dahil edebiliriz. Geliştirdiğimiz yazılımı test ettikten sonra görülen sıkıntılar düzeltilir ve müşteriye teslim etmeye hazır hale getirilir. Test kodlama sürecindeki en önemli adımdır.

5) BAKIM VE DESTEK: Yazılımın teslim edildiği müşteriyle belli bir süre için anlaşma yapılır. Anlaşma sürdüğü sürece teslim edilen projenin bakımını yapmak sorumluluktur. Döngü teslim etmeyle son bulmaz. Çeşitli hata giderme ve güncelleme işlemleri bu aşamaya dahildir.

Yazılım yaşam döngüsü yaklaşımının planlı çalışmayı sağlamak, hataları görüp ilerlemeyi kolaylaştırmak, modellemeler ile normal yaşama adapte edilmeyi kolaylaştırmak gibi avantajları vardır. Ama küçük projelere uygun değildir. Çünkü minik teslim sürelerini fazla uzatır. Bu da maliyeti arttırır.

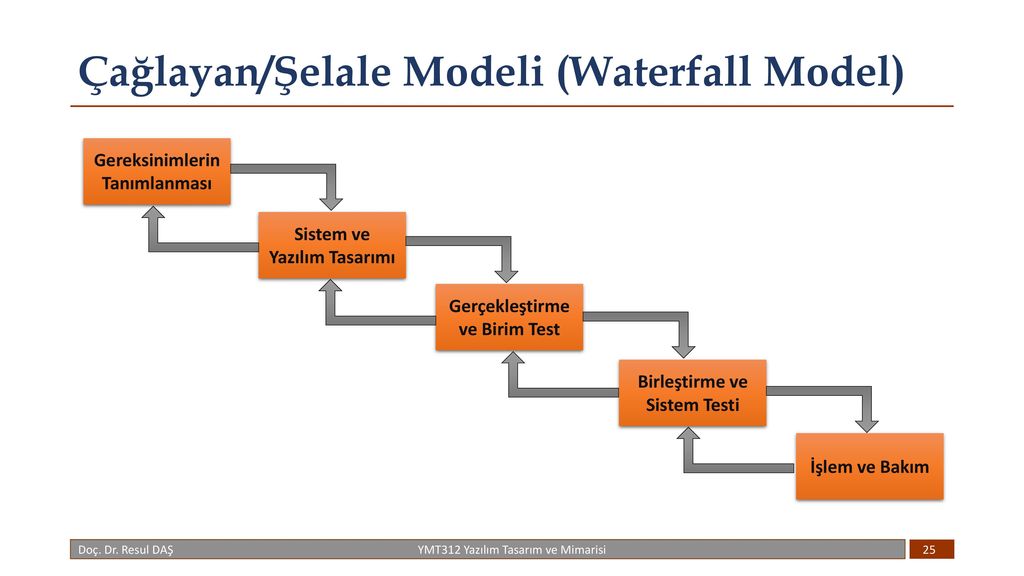
Yazılım yaşam döngüsünün en iyi şekilde çalışması için yazılım geliştirme süreç modelleri (processes) kullanılır. Bu modeller; düzenleyici süreç modelleri, birleşik süreç ve çevik yazılım süreci olarak ayrılırlar.

**Düzenleyici Süreç Modelleri** KODLA VE DÜZELT (CODE and FİX): Konu üzerinde acemiyken, öğrenirken veya araştırma yaparken kullanılacak en iyi modeldir. Planlama aşaması atlanarak direk uygulamaya geçilerek zaman tasarrufu sağlanır. Ancak bu tasarruf edilen zaman büyük projelerin sonraki aşamalarında bize çok fazla maliyet çıkararak geri dönebilir. Büyük projelerde tercih edilmez.

GELİŞİGÜZEL MODEL: Bir model olarak tanımlanamaz denilebilir. Adında da görüldüğü üzere gayet gelişigüzel bir modeldir. Eski yıllarda kullanımı daha yaygındır. (60’lı yıllarda diyebiliriz.). Takibi zordur. Plansızdır. Kodla ve düzelt modeliyle benzer yerlerde kullanılabilir. Acemiler, geliştirme aşaması, öğrenci projeleri vb. yerlerde bu model görülebilir.

BAROK MODEL: Döngünün temel adımları doğrusaldır ve geri dönüşler belirsizdir. Dokümantasyon test aşamasından sonra yapılır ve ayrı bir adım olarak görülür. Gerçekleştirim aşamasına da çok önem verdiği için kullanımı yetmişlerde daha yaygın olan ama günümüzde pek kullanılmayan modeldir.

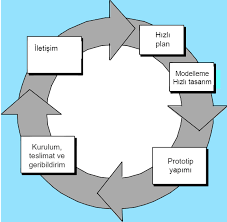
ÇAĞLAYAN (WATERFALL/ ŞELALE) MODELİ:



Şekil 2: Çağlayan Modeli

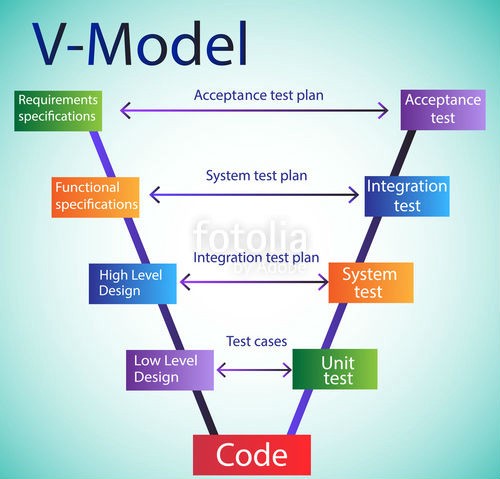
Yaşam döngüsü adımları doğrusaldır. Geri dönüş olmaması için her adım için gerekilen önem verilmeli ve hatasız ilerlenmelidir. Her adım iyice incelenerek ilerletildiğinden projenin bitmesi uzun zaman alır. Bu duruma müşteri/patron sorun olarak bakabilir. Geri dönüp hataları düzeltmek de aşırı maliyetlidir. Küçük ve gereksinimleri bariz belli olan projelerde kullanılması daha doğru olur. Bütün aşamalar tek tek belgelenir ve o şekilde ilerlenir. Belgeleme ve test olmamış aşama tamamlanmış sayılmaz. En çok analiz ve tasarım aşamasında zaman harcanır. Geri dönüşler tanımlıdır. Değişimlere açık değildir. Ancak olumsuzluklarına rağmen diğer modellere örnek olduğu için temel bir modeldir. Kullanımı gün geçtikçe azalmaktadır.

PROTOTİPLEME: Hızlı plan, hızlı tasarım, hızlı kodlama bu modelin anahtar kelimeleri denilebilir. Buradan da belli olduğu üzere hız çok önemlidir. Projeyi geliştirenler ve müşteriler/kullanıcılar bir masada toplantı yaparak gereksinimlere, yapılacak işlere ve tasarımlara karar verirler. Bu kararlara göre kullanıcıya bir örnek çıkartılır ve kullanıcı beğenirse anlaşma yapılır. Geliştirici ve müşteri arası bilgi alışverişi ile yazılım oluşturulur. Düzeltme adımı atlanmamalıdır.



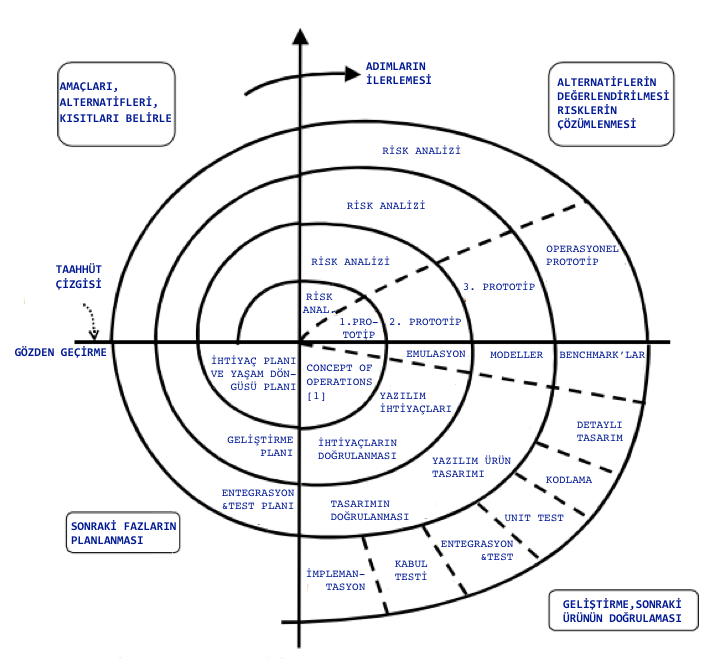
Şekil 3: Prototipleme

V MODELİ: Waterfall (Şelale) modelinin daha gelişmişi olarak bakabiliriz. Şelale modeline doğrulama ve onaylama aşaması eklenmiştir. Takibi ve kullanımı kolaydır. Gereksinimlerin daha net olduğu ve belirsizliğin az olduğu projelere daha uygundur. Adımlar doğrusal ilerlemez. Kodlamadan sonra yukarı doğru şekillenir. (Şekil 4) Risk çözümlemesi yoktur. Sonuca düz bir çizgide ulaşılmaz. Unit test aşaması kodların doğruluğunu kontrol eder. Sistem test aşaması sistem tasarımını kontrol eder. Entegrasyon testleri mimari tasarımı kontrol eder. Müşteri testleri ise gereksinim analizini kontrol eder. Temel çıktıları; kullanıcı modeli, mimari model ve gerçekleştirim modelidir. Kullanıcı modeli, geliştirme sürecinde kullanıcı ile ilişkileri tanımlar ve sistemin nasıl istendiğini ortaya çıkarır. Mimari model, sistem tasarımını inceler. Gerçekleştirim modeli, kodlama kısmı ile ilgilidir.



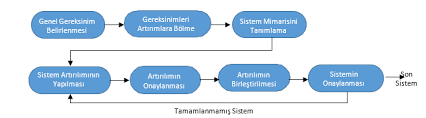
Şekil 4: V Modeli

HELEZONİK MODEL (SPİRAL MODEL): Sarmal modelin anahtar kelimeleri prototip üretme ve risk analizidir. İlk önce projenin riskli kısımlarına çözüm üretir. Riski minimum seviyede tutarak başarılı bir yazılım geliştirir. Bu sayede maliyet ve zaman açısından avantaj sağlanır. Kodlamanın erken başlaması yazılımı geliştiren için avantajdır. Kullanıcı sürece erken katılır. Bu da sonradan oluşacak istenmeyen durumların veya sonradan kullanıcıdan gelecek değişiklik talebinin önüne geçer. Kullanıcı ve geliştirici için kolaylık sağlar. Önceden geliştirilmiş yazılım ürünleri üzerinde çalışılan projeler için kullanılır. Küçük projeler için uygun değildir.



Şekil 5 :Helezonik (Spiral) Model

ARTIMSAL GELİŞTİRME MODELİ: Bu modelde çekirdek bir yapı korunur ve üstüne parça parça yeni kısımlar eklenerek değişiklikler sağlanarak son ürüne ulaşılır. Çekirdek yapı dediğim şey en temel gereksinimleri yerine getirerek oluşturulmuş yazılımdır. Bu modelde teslim tek seferde yapılmaz. Çekirdek yapı erken teslim edilir. Eklenecek kısımlar sonraki kısımlarda ele alınır. Bu şekilde parça parça gidildiği için yazılım komple çökmez. En önemli gereksinimler en başta karşılandığı için en çok onlar test edilmiş olur. Yinelemeli bir modeldir. Az çalışanla çok iş yapmayı sağlayabilir. Bu model bir böl ve yönet yaklaşımıdır. Kullanım ve üretim aynı anda yapılır. Genelde deneyimli insanlarla çalışmayı gerektirir.



Şekil 6: Artımsal Geliştirme Modeli

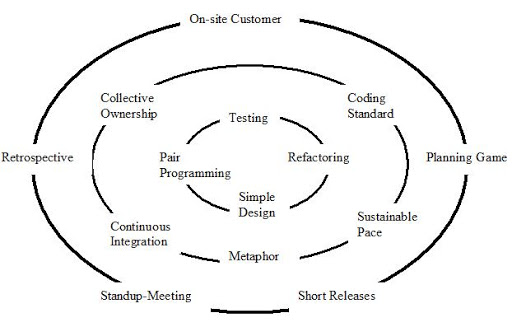
**Birleşik Süreç Modeli**

Yazılım geliştirme süreçlerinin en güzel özellikleri bir araya getirilerek oluşturulmuş modeldir

**Çevik Yazılım Süreci**

Çevik yazılımın genel özellikleri; basitlik, günlük çalışma, haftalık program, müşteri/çalışan iletişimi, modül bazlı çalışma ve modül bazlı test diyebiliriz. Çok sık kullanılan bir metottur. Hızlı olmak gereklidir. Müşteriyi memnun etmek ve projenin sürekliliği önemlidir. Takım motivasyonu önemlidir.

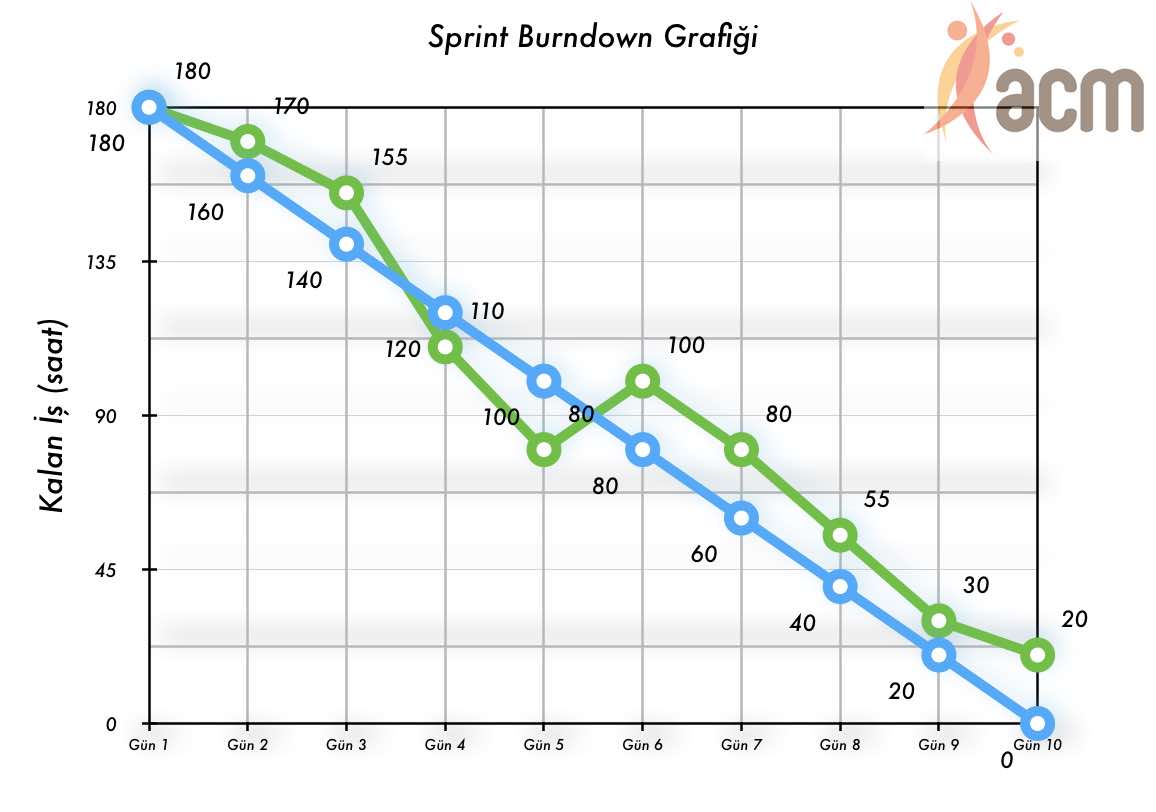
EXTREME PROGRAMMING (XP): En popüler çevik yazılım süreçlerinden biridir. Grup içi iletişime çok önem verir. Çalışmanın başlarında elle tutulur bir şeyler elde edilebildiğinden hatalar erken safhada görülebilir ve çözülebilir. Artımsal yaklaşım sayesinde hızlıca proje planı yapılır ve süresi tahmin edilir. İletişim en önemli kısmıdır. Yanlış anlaşımları en aza indirmek hatta sıfırlamak amaçlandığı için iletişim mutlaka yüz yüze olmalıdır. İkinci önemli kısmı ise basitliktir. XP ile yazılacak programda gereksiz hiçbir kısım olmamalıdır. En basit en iyidir. Sistemin geri dönüşümlü olması çok önemlidir. Dördüncü ve en zor kısmı ise cesarettir. Korku yazılımı başarısızlığa iter. Başarısızlıktan korkulmamalıdır. XP pratikleri; planlama oyunu, ekipte müşteri, önce test, basit tasarım, çiftli programlama, sürekli entegrasyon, kısa aralıklı sürümler, yeniden yapılandırma, ortak kod sahiplenme, metafor, kodlama standardı ve haftada 40 saattir.



Şekil 7 XP Modeli Pratikleri

SCRUM: Bir diğer popüler çevik yazılım metodudur. Daha karmaşık projeleri yönetmek için kullanılır. Parçalara ayırarak tekrara dayalı bir yöntem izler. Esnek bir yapısı vardır. Diğer çevik modellerde de olduğu gibi iletişim ve takım çalışması oldukça önemlidir. Prensipleri; şeffaflık, denetleme ve uyarlamadır. Projenin ilerleyişi herkese açık, şeffaf olmalıdır. İlerleyiş düzenli olarak denetlenir. İlerlemenin birincil ölçüsü çalışan bir yazılımdır. Değişikliklere açıktır, uyarlanır.

BİLEŞENLER Product Backlog: Gereksinimlerin olduğu listedir. Proje sürecinde ekleme çıkarma yapılabilir. Product Backlog Item: Listedeki gereksinimlerin teker teker ismidir. Sprint: Projenin ayrıldığı küçük parçalara verilen addır. Sprint Backlog: Gereksinimlerin sprint içinde öncelik sırasına göre dizilmesidir. Scrum Board: Sprint içinde yapılacak maddeler buraya yazılır. Burndown Chart: Şeffaflığı sağlar. Yatay ekseninde sprint günleri, dikey ekseninde sprintte kalan işi gösteren grafiktir.



Şekil 8 Burndown Chart Örneği ROLLER Product Owner: Geliştirme takımı ve müşteri arası iletişimi sağlar. Sprinti iptal edebilir. Product backlog oluşturur ve proje özelliklerini tanımlar. Geliştirme Takımı: Sprintteki tüm işleri tamamlayabilecek niteliklere sahip kişilere denir. Herkes her işi yapabilir. Belli bir görev dağılımı yoktur. Kendi kendilerini yönetirler. Scrum Master: Takımın en rahat çalışabileceği ortamı hazırlamak görevidir. Yönetici değildir. Chicken Roller: İşleyişte aktif yer almayan kişilere verilen addır.

TOPLANTILAR Sprint Planning: Product owner, scrum master ve geliştirme takımı katılır. Product backlogdaki gereksinimler küçük parçalara ayrılır ve herkes kendi hızına göre bir görevi üstlenir. Daily Scrum: Her gün yapılan 15 dakikalık toplantılardır. Kimse davet beklemez. Dün neler yaptığını anlatır. Sorun varsa scrum master çözer. Koşul ne olursa olsun bu toplantılar mutlaka yapılır. Sprint Review: Her sprintin sonunda yapılır. Amacı yazılımın gereksinimlere göre yapılıp yapılmadığını kontrol etmektir. Sprint Retrospective: Scrum takımının kendini geliştirmesi için oldukça yararlı bir toplantıdır. Sprint boyunca yapılan işler, doğrular ve yanlışlar tartışılır. SCRUM NEDEN POPÜLER?

Scrum en popüler çevik yazılım şeklidir. Çünkü oldukça şeffaftır. Kullanıcı/müşteri ve çalışanlar net bir şekilde sürekli takibini yapabilirler. Takım düzenli aralıklarla nasıl verimli olabileceğini tartışır. Bu nedenle verim fazlasıyla yüksektir. Ayrıca çalışanlar kendini sürekli geliştirebilir. Baş ilkesi basitliktir. Bu da projelerin gereksizce uzamasına engel olur. Projeler yeterince kısa bir sürede geliştirilir ve teslim edilir. Müşteri memnuniyeti temellidir. Müşteri gereksinimleri çok iyi dinlenir ve anlanır. Yanlış anlaşılmayı en aza indirger. Yapılan düzenli toplantılar son anda karşılaşılacak kötü sürprizleri engeller.

**MODELLERİN KARŞILAŞTIRILMASI**

Maliyet açısından modelleri karşılaştırmak gerekirse; kodla ve düzelt, prototipleme, artımsal geliştirme düşük maliyetliyken helezonik model, çağlayan modeli ve v modeli maliyetlidir. Çevik yazılım modelleri ise oldukça maliyetlidir.

Getirdiği başarı açısından kıyaslarsak, kodla ve düzelt modeli ve çağlayan modelinin başarısı yetersizdir. Prototipleme modeli ve v modelinin başarısına orta seviye diyebiliriz. Helezonik model ve artımsal geliştirme modelinin başarısı yüksek, çevik modellerinin başarısı ise fazlasıyla yüksektir.

Yeniden kullanabilme özelliklerini kıyaslarsak; artımsal geliştirme modeli, çevik modeller, prototipleme ve helezonik model yeniden kullanılabilir. Ancak kodla ve düzelt modeli, çağlayan modeli ve v modelinden bu pek mümkün değildir.

Esneklik açısından karşılaştırırsak; artımsal geliştirme modeli, çevik modeller, prototipleme ve helezonik model fazlasıyla esnektir. Ancak çağlayan modeli ile kodla ve düzelt modelini katı olarak niteleyebiliriz.

Barok model ve gelişigüzel model ise zaten geçerliliğini yitirmiştir.

**KAYNAKÇA**

1) <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> Erişim tarihi : 20.03.2021 2) <https://medium.com/@secilcor/scrum-nedi%CC%87r-6a4326951dd8> Erişim tarihi : 26.03.2021 3) <https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/> Erişim tarihi : 22.03.2021 4) <https://medium.com/@ahmetuyar/extreme-programming-xp-nedir-ddc003a515c4> Erişim tarihi : 26.03.2021 5) <https://blog.burakkutbay.com/cevik-agile-yazilim-gelistirme-metodu-nedir.html/> Erişim tarihi : 25.03.2021