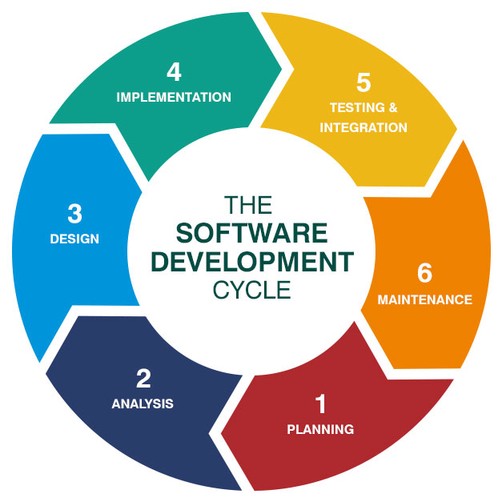
YAZILIM YAŞAM DÖNGÜSÜ VE MODELLERİ

Yazılım yaşam döngüsü, (Software Development Life Cycle SDLC) yazılımın hem üretim hem de müşteri tarafından kullanımı süreçlerinde devam eden, herhangi bir yazılımın, üretim aşaması ve kullanım aşaması birlikte olmak üzere geçirdiği tüm aşamalar biçimine yazılım yaşam döngüsü denir. Bu model sayesinde planlı çalışmak kolaylaşacak ve daha kaliteli ürünler ortaya çıkacaktır. Bu döngü planlama (planning), analiz (analysis), tasarım (design), gerçekleştirme(implementation) test ve son olarakta teslim ve bakım (Maintenance) aşamasından oluşur.



**1. Planlama Aşaması:** Döngümüzün ilk aşamasıdır. Bu aşama yapılacak ürünün ana hatlarının ortaya çıktığı aşamadır. Üretilecek olan ürün için müşteriden müşterinin istekleri alınır ve plan yapılır böylelikle süreç başlamış olur.

**2. Analiz Aşaması:** Yazılım işlevleri ile gereksinimlerin ayrıntılı olarak çıkarıldığı aşamadır. Bunlara bağlı olarak sistem gereksinimleri netleşir ve buna bağlı istekler hazırlanır. Kısacası projenin tüm detayları ortaya çıkartılır.

**3. Tasarım Aşaması:** Analiz aşamasının sonucunda ortaya çıkartılan proje detayları baz alınarak proje gerekli durumlarda bileşenlerine ayrılır. Diğer bir değişle ürünün özellikleri, yetenekleri ve arayüzlerinin belirlenmesidir.

**4. Gerçekleştirme ve Test Aşaması:** Ürünü somut olarak ortaya çıkardığımız aşamadır. Tasarım aşamasından gelen gereksinimler doğrultusunda kodlama yapılır. Yazılım geliştirilmesi tamamlandıktan sonra müşteriye sunmadan önce, test ekibi tarafından beta testlerinin gerçekleştirilir. Bu noktadan sonra herhangi bir analiz işlemi yapılmamalı, proje, tasarım aşamasında oluşturulan planlama çerçevesinde ilerlemelidir.

**5. Teslim ve Bakım Aşaması:** Teslim edilen projenin artık bakım aşaması proje yaşamı boyunca sürecek şekilde başlamış olur. Yazılımın iyileştirilmesi ve yeni işlevlerin eklenmesi süreçleridir. Bu süreç zarfında kullanıcılardan gelen geri bildirimler sayesinde bu istekler gerçekleştirilmektedir.

**YAZILIM YAŞAM DÖNGÜSÜ MODELLERİ**

Yazılım geliştirme yaşam döngüsünde, süreçlerin geliştirme aşamasında, sürecin düzeni ve nasıl uygulanacağını ifade eden modellerdir. Karmaşıklık düzenini azaltır ve böylece oluşacak riskleri önler.

**Gelişigüzel Model:** Çok eski ve basit bir modeldir. 1960’lı yıllarda kullanılmıştır. Geliştiren kişiye bağımlıdır ve belli bir tekniği-yöntemi yoktur.1960 yıllarda kullanılıp popülerliğini yitirdiği için tam anlamıyla bir modeldir diyemeyiz.

**Barok Modeli:** Barok modeli, 70’li yılların ortalarında kullanılmaya başlanmıştır. Günümüzde kullanımı yitirmiştir. Yazılım yaşam döngüsü temel adımları doğrusal bir şekilde ele alınır. Gerçekleşme evresine çok fazla ağırlık veren bir model olduğu için kullanımı günümüzde tavsiye edilmemektedir.

**Çağlayan Yaşam Modeli**: Aynı zamanda şelale veya geleneksel model olarak da adlandırılır. Şelale yönteminde yazılım geliştirme süreci analiz, tasarım, ​​ kodlama, test, sürüm ve bakım​​ gibi aşamalardan oluşur. ​​Geleneksel yazılım yöntemlerinde bu safhalar şelale​​ modelinde olduğu gibi işler. Her aşama, ​​ başlangıç noktasında bir önceki aşamanın ürettiklerini bulur. İyi tanımlı projeler ve üretimi az zaman gerektiren yazılım projeleri için uygun bir modeldir. Geleneksel model olarak da bilinen bu modelin kullanımı günümüzde giderek azalmaktadır.

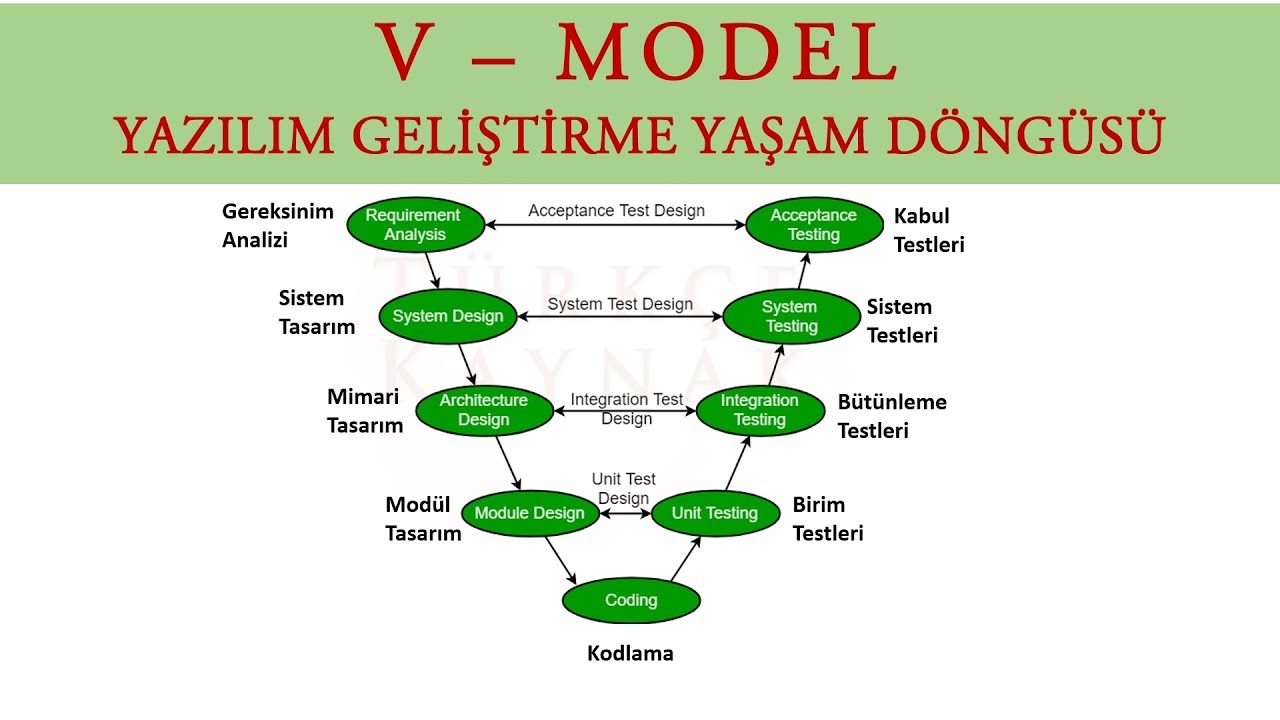


**V Süreç Modeli:** V süreç Modeli, Şelale modelinin belirli düzenlemeler yapılmış bir uzantısıdır. Her temel yazılım geliştirme aşamasına paralel olarak bir test aşamasını içerir. V şeklinin sol tarafı üretimi, sağ tarafı ise testi temsil eder. Bu modelde proje takibi kolaydır. Modelin kullanımı basittir. Aşamalar arası tekrarlamalar bulunmaz. V süreç modelinin kullanıcı modeli, mimari model ve gerçekleştirim modeli olarak adlandırılan üç alt modeli vardır.

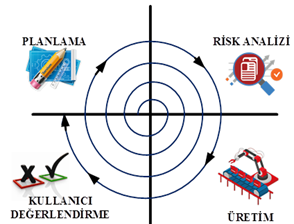
Kullanıcı Modeli; Geliştirme sürecinin kullanıcı ile olan ilişkilerini tanımlar

Mimari Model: sistem tasarımı ve oluşacak alt sistem ile tüm sistemin sınama işlemlerine ilişkin işlevler yer alır.

Gerçekleştirim Modeli: yazılım modüllerinin kodlanması ve sınanmasına ilişkin fonksiyonlar yer alır.



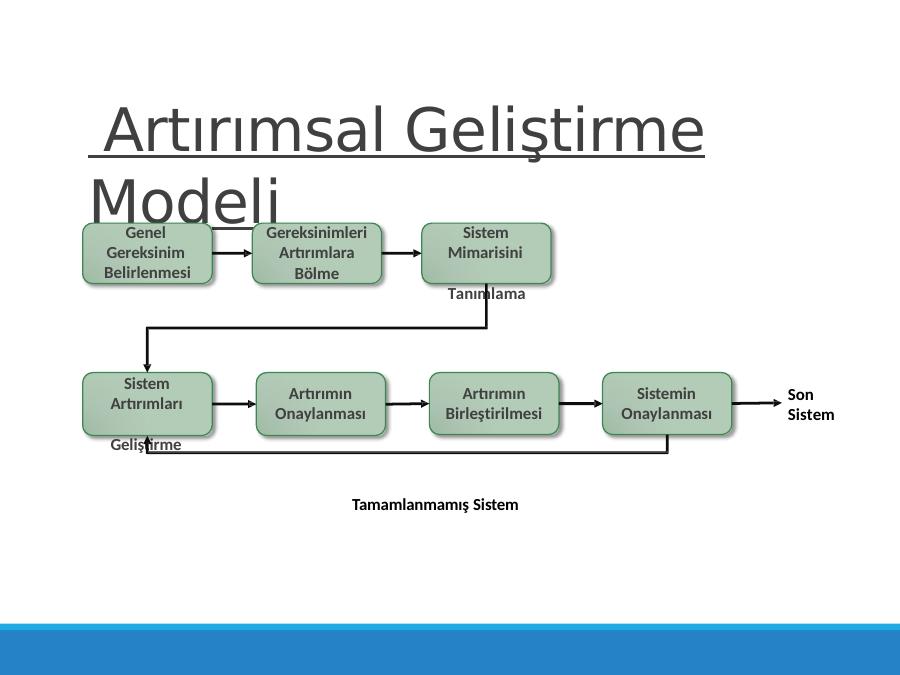
**Helezonik Model:** Diğer adıyla Sarmal model, yazılım yaşam döngüsü devam ederken önceki safhalara dönmenin zorunlu olduğu bir modeldir. Şelale modelinde yok sayılan riskleri göz önünde bulundurur, Proje çevrimlere ayrılır ve her bir çevrimin riskleri ayrı ayrı ele alır bu yüzden özellikle güvenlik yazılımlarının oluşumunda bu model kullanılmaktadır. Kullanıcı katkısı, mühendis bakış açısı ve yönetici bakış açısının bir arada olduğu projelerde kullanılır.



**Kodla ve Düzelt Yaşam-Döngü Modeli**: Genellikle yüzlerce satırlık çok uzun kodlarda kullanılır. Yazılım geliştirmenin en kolay yoludur. Bu yüzden küçük firma ve kuruluşlarda en çok kullanılan yöntemdir. Aynı zamanda çok maliyetli bir sistemdir. İlk olarak bir ürün ortaya çıkarılır. Daha sonra bu ürün sürekli olarak geliştirilir.

**Evrimsel Geliştirme Modeli:** İlk tam ölçekli modeldir. Coğrafi olarak birbirinden uzak olarak bulunan birimlerin birlikte hareket edebilmesi için bulunmuş bir yöntemdir. İlk olarak bir ürün piyasaya sürülür. Daha sonra bu ürün test edilir ve gerekli güncelleme ve düzenlemeler yapılır. Bu ürünün kalitesi ve işlevselliği ilk ürüne bağlıdır.

**Artırımlı Geliştirme Süreç Modeli:** Ürünü tek seferde teslim etmek yerine her yazılım sürümünde yeni işlevler eklenerek teslim edilen ve çağlayan modeli ile Evrimsel geliştirme arası bir modeldir. Kullanıcı gereksinimleri önceliklerine göre parçalanıp yeni sürümlerde parça parça eklenerek devam edilir. Bu süreçte ürün bir tarafta kullanılırken diğer kısımda ise hala ürünün gelişimi devam etmektedir. Üretimi uzun sürecek ve eksik gereksinimlere rağmen çalışabilecek projelerde uygun bir modeldir.

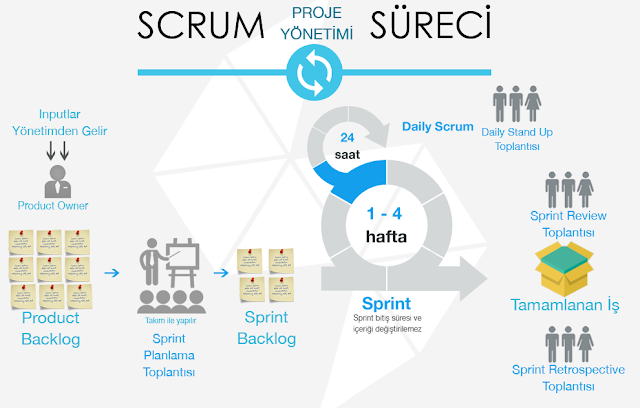


**Çevik Yazılım Modeli:** Geleneksel yazılım geliştirme metotlarına tepki olarak 1990’lı yıllarda ortaya çıkmıştır. Genellikle takım çalışmasını içeren çevik modeller özellikle müşteri ihtiyaçlarını ön planda tutar. Kullanıcı deneyimine ve değerlendirmesine odaklanır. Bu yöntemin amacı, geliştirme sürecindeki bazı yüklerden kurtularak sonuca daha hızlı ulaşıp, değişen ihtiyaçlara daha erken cevap vermektir.

**SCRUM:** Scrum 1990lı yıllarda bulunan bir proje yönetme aracıdır. Çevik yazılım geliştirme yöntemlerinden bir tanesi olan scrum kompleks yazılım süreçlerinin yönetilmesinde kullanılır. Kompleks yapıdaki işleri küçük birimlere bölerek geliştirmeyi ön gören bu metot, karmaşık ortamlarda adım adım çalışan yazılım geliştiricileri için epey bir uygundur.

**Scrum neden günümüzde de popülerdir?**

Scrum, rol örtüşmesini hesaba katan çok disiplinli bir metodolojidir; bu özellik, tüm ekip üyeleri tarafından oynanan bireysel rollerin daha iyi anlaşılmasını ve takdir edilmesini olanak sağlar. Müşteri ile orta noktada buluşarak ürünü en verimli hale getirmeyi amaçlar. İletişim sayesinde hatalar azalır. Müşteriler sayesinde devamlı geri bildirimler alarak, çok hızlı ve kaliteli projeler geliştirebilir. Günlük yapılan kısa toplantılar ile de kontrolleri sağlayan metot hata ve takım arasındaki iletişim kopukluklarına da engel olur. Bu yüzden scrum metodu yazılım geliştiriciler açısından oldukça popüler ve kullanışlıdır.



Mehmet Salih İnan 210601690

Linkledn hesap profili: <https://www.linkedin.com/public-profile/settings?trk=d_flagship3_profile_self_view_public_profile>

Githup Profili: <https://github.com/Msinan01>

Medium Profil: <https://medium.com/@ms.inan01>