YAZILIM YAŞAM DÖNGÜSÜ VE MODELLERİ

Yazılım bir üründür ve her ürünün bir ömrü vardır. Bir yazılımın geliştirilmesi ve bakım yapıldığı süre boyunca izlediği adımlara yazılım yaşam döngüsü denir. Yazılımın yaşam döngüsü birden çok temel aşamadan meydana gelmektedir. Bu aşamalar şunlardır: Gereksinim, analiz, tasarım, gerçekleştirme ve bakım. Yani planlama evresinden kullanılmaya başlanarak süren bir döngüdür.

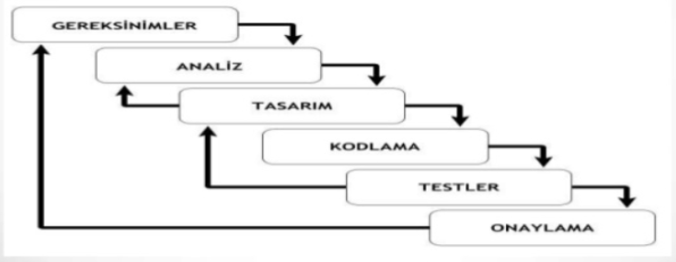
**Gereksinim**: Müşterinin istekleri ve gereksinimlerinin elde edildiği ve yapılabilirliğinin konuşulduğu aşamadır. **Analiz-Çözümleme**: Sistem gereksinimlerinin ve işlevlerinin ayrıntılı bir şekilde incelendiği aşamadır. **Tasarım**: yazılımın temel yapısının atıldığı aşamadır. Belirlenen gereksinimlere yanıt vermesi gerekir. **Gerçekleştirme:** Modüller kodlanır, birleştirilir, test edilir ve kurulum çalışmalarının yapıldığı aşamadır. **Bakım:** Teslim ettikten sonra bilinen hataların giderildiği ve yeni gereksinimlerin eklendiği aşamadır.

**1. Yazılım Döngü Modelleri**

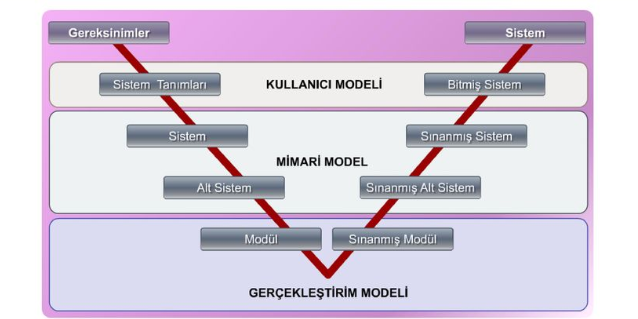
**1.1. Gelişigüzel Model**: Herhangi bir yöntem ya da model mevcut değildir. Kişiye özel, kişiye bağlı yazılım geliştirme şeklindedir. Yazılımın izlenebilirliği ve bakım yapılması zordur. Genellikle basit programlama içeren ve çoğunlukla tek bir kişinin üretim yaptığı yöntemdir. (Günümüzde artık kullanılmamaktadır.)

**1.2. Barok Modeli**: 1970’li yıllarda ortaya çıkmıştır. Yazılım yaşam döngüsünün temel adımlarını doğrusal bir şekilde ele alıp geliştirir. Aşamaların arasında geri dönüşler yoktur yani bir döngü değildir. Belgelendirme farklı bir süreç olarak işlenir ve yazılım geliştirme ve test aşamalarının sonrasında yapılır. (Günümüzde artık kullanılmamaktadır.)

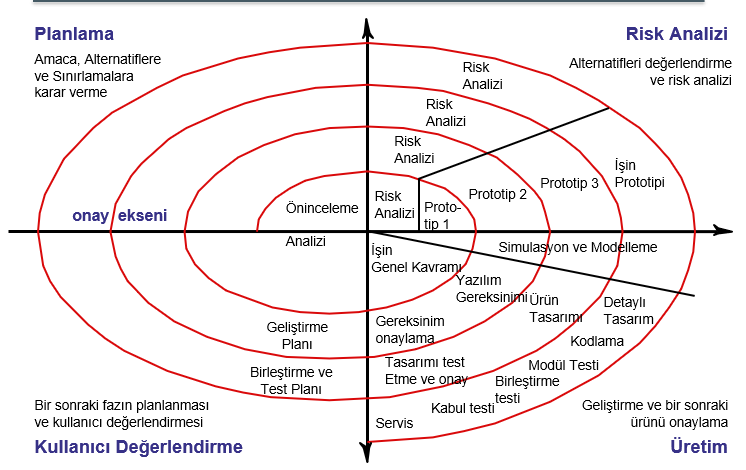
**1.3. Şelale Yaşam Döngü Modeli**: Geçmişte en çok kullanılan, en popüler modeldi. Diğer güncel modellerin temeli sayılır. Geleneksel yazılım geliştirme modeli olarak da bilinir. Her aşama en az bir kere tekrar edilir. Belgelendirme sürecin içindedir. Adım adım ilerlenir ve aşama atlanmaz. Az zaman gerektiren projelerde kullanılır ve projenin iyi tanımlanmış olması gerekir. Günümüzde kullanımı azalmaktadır. Bazı dikkat edilmesi gereken konular vardır. Analiz aşamasındaki tüm detayların tasarıma yansıması için müşterinin gereksinimlerinin iyi belirlenmiş, detaylı belirlenmiş olması gerekir. Tasarım aşaması da projenin tüm gereksinimlerini karşılayacak düzeyde olmalıdır. Projelerin yineleme gerektirdiğinden, yazılımın kullanıcıya ulaşmasının uzunluğundan ve gereksinim tanımlamasının eksik veya yetersiz yapılmasından ötürü, yanlışların düzeltilmesi ve eksiklerin giderilme maliyetleri yüksektir. Kullanıcı sürecin içinde yer almadığı için yazılım sonunda oluşan geri dönüşlerle maliyetleri yüksek ölçüde arttırır.



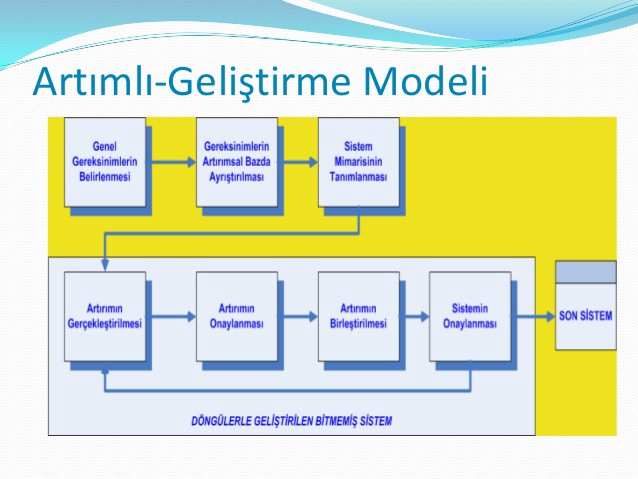
**1.4. V Süreç Modeli**: Bir taraf üretim diğer taraf sınama (test) işlemleridir. 3 temel çıktı vardır. Kullanıcı modeli, geliştirme sürecinin kullanıcı ile iç içe olduğu planları ortaya çıkarır; Mimari model, sistem tasarımı ve oluşacak alt sistem ile tüm sistemin sınama işlemlerine ilişkin görevler; gerçekleştirim modeli, kodlama aşaması ve sınanma uygulamaları. Belirsizlik az ve gereksinimlerin belirgin olduğu Bilgi Teknolojileri projeleri için uygun bir modeldir. Model kullanıcıyı projenin içine katar.



**1.5. Helezonik (Spiral) Model**: bu modeli diğerlerinden ayıran en önemli unsur risk analizinin ön planda olmasıdır. Bu sayede hataları erken tespit edip düzeltme imkânı tanıyabilir. Kullanıcı projenin içinde olduğundan bu modelde de sorunların azalmasına olanak sağlayabilir. Doğrudan gereksinim, analiz, tasarım, test gibi fazlar yoktur. Yinelemeli artımsal yaklaşım vardır. Bu modelin kullanım avantajları vardır. Bunlardan biri kullanıcı katkısı, üretim boyunca kullanıcının işin içinde olması ürünü sınamak için bir fırsattır ve olası hataları erkenden engeller. Yönetici bakışı; projedeki yöneticiler, proje süresince karşılaşılan sorunlar veya takım içi ilişkilerle daha yakından daha kolay izleme planlaması yapılır. Son olarak mühendis bakış açısı, ürünün sınanması ve sürülmesi daha erken başlar.



**1.6. Artımsal Geliştirme Süreç Modeli**: Tek seferde teslim yerine kullanıcın önceliğine göre sistemi parçalara böler. Bu parçalar her bittiğinde kullanıcı kullanmaya başlar. Ara ürünler her bir aşamada bir öncekinin üstüne yeni şeyler katarak çıkartır. Yani bir yandan üretim yapılırken diğer yandan kullanım devam eder. Uzun zaman alabilecek ve sistemin eksik işlevsellikle çalışabileceği türdeki projeler bu modele uygun olabilir. Müşteri sistemi erkenden kullanmaya başladığı için sistemin işlevselliği erken aşamalarda ortaya çıkar. Sistemin başarısız olma ihtimali azalır ve olası bir hata durumunda projenin tümden batmasını engeller.



**1.7. Kodla ve Düzelt Yaşam Döngü Modeli**: Küçük programlar için kullanılabilir. Direkt olarak yazılım ürünü gerçekleştirilir. Sistem son haline gelene kadar devamlı geliştirilir. Bakım safhası vardır ama dokümantasyon olmadığı için bu safha çok zordur. Emeklilik safhası da vardır.

**2. Çevik Yazılım Geliştirme**

Çevik modeller yazılım projelerinde ekip içi iletişimi ve başarı oranlarını artırmak için ortaya çıkmıştır.

**2.1. Extreme Programming (XP):** Kent Beck tarafından 1999 yılında kurulmuştur. Grup içi iletişimi ön planda tutan ve geri dönüşlerin daha fazla olmasına imkan sağlayan bir yazılım geliştirme yöntemidir. 4 temel maddeden oluşur: Basitlik, Geri Dönüş, Cesaret, İletişim.

**Basitlik:** Üretilen her projeyi, her kodu, her tasarımı basit yapılmasını gerektirir.Uzun dokümantasyondan kaçılır **Geri Dönüş**: Geri dönüşler ile oluşabilecek hataları görüp düzeltme imkanı tanır. Müşteri ile yazılım ekibi sürekli iletişim halindedir. **İletişim:** takım üyelerinin iletişimi projelerde olması gereken bir unsurdur. XP bunu çözmek için çalışmaktadır. **Cesaret:** ne iş yapılırsa cesur olunmalıdır. Projelere korkmadan başlamak gerekir. Gerekirse bir kod tamamen silip yeniden yazılır

**2.2. Scrum**: 1990’ların ortalarında geliştirilen, çevik yazılım geliştirme metotlarının uygulanabilindiği bir proje yönetim yaklaşımıdır. Karmaşık işleri bölerek geliştirir, her parçaya sprint adı verilir. Bu yöntem karmaşık ortamlarda adım adım geliştiren ekipler için uygundur. Gereksinimlerin belirsiz olduğu projeler için en uygun yöntemdir. 3 temel kavram vardır: Roller, Toplantılar, Araçlar.

**Roller**: Ürün sahibi, scrum yöneticisi ve scrum takımından oluşur. Ürün sahibi proje beynidir. Scrum yöneticisi, scrum takımını kontrol eder. Scrum takımı ise 5-9 kişiden oluşan ve birbiri ile sürekli iletişim içinde bulunan bir yazılım geliştirme ekibidir.

**Toplantılar**: Scrum için toplantılar çok önemlidir. Her gün scrum toplantıları yapılır. Sprint planlama, sprint gözden geçirme, günlük scrum toplantısı olmak üzere ayrılır.

Sprint planlama: gereksinim listesinin çıkarıldığı toplantıdır, uygun dağıtımların yapılması ve bunlar için takımlar belirlenir. Risk değerlendirmesi yapılır. Dağıtım, geliştirme ve pazarlama maliyetleri hesaplanır. Sprint gözden geçirme: Her sprint başında planlama toplantısı yapılır. Ürün sahibi ile takım ürün gereksinimlerini belirler ve sprint gereksinim listesi oluştururlar. Takım ürün sahibi tarafından belirlenen gereksinimlerin ne kadarını yapacağını belirler. Günlük scrum toplantısı: Bu toplantılar 15 dakika civarı olup tüm takımın dahil olduğu genellikle sabahları yapılır. Takımın ilerleyişini ve öndeki sorunları görmek için idealdir.

**Araçlar:** ürün gereksinim dokümanı oluşturulur. Bu dokümanda proje boyunca yapılacak şeyler yazılır. Sprint dokümanı oluşturulur. Her sprintin ona uygun ayarlanmasını sağlar. Sprint kalan zaman grafiği ise projenin neresinde olunduğunu ve proje bitimine ne kadar zaman kaldığını belirlemek için hazırlanır.

Scrum günümüzde en çok kullanılan yazılım geliştirme yöntemidir. Bunun nedenleri zamandan ve paradan tasarruf etmesi, yüksek teknolojiler ve son gelişmelere kolaylıkla uyabilmesi, karmaşık ve gereksinimlerin belirsiz olduğu projelere uyması, ekip içi iletişimin yüksek olması ki bu sebepten ötürü hatalar erkenden fark edilir ve düzeltilir, yinelemeli olması, değişen gereksinimlere hızlı tepki verebilmesi ve böl ve fethet yaklaşımı içinde olması yani projeyi bölerek, parçalara ayırarak çalışmasıdır.

**Hangi yazılım geliştirme modeli daha kullanışlı**

Gelişigüzel ve barok modeli günümüzde kullanımını yetirmiştir. Yinelemeli olmamaları ve dokümantasyonu işin dışında tutmaları başlıca nedenleridir.

Şelale modelinin kullanılması ve uygulanması basit, iş bölümü ve planlaması detaylıdır. Küçük ve gereksinimleri belirli projelerde kullanılır. Artık eskisi gibi yaygın kullanılmamaktadır. Kullanıcı ile iletişim az olması, büyük projelerde yetersiz kalması ve değişimlere kapalı olması gibi sorunlar yüzünden eskisi gibi kullanılmamaktadır.

V Süreç Modeli ise şelale modelinin gelişmiş halidir. Takip edilmesi ve kullanımı kolay olup onaylama işlemlerinin erkenden yapılması avantajlarıdır. Fazlar arası tekrarlamalar olmadığı için ve risk aşamalarının eksikliği kullanımını azaltmıştır.

Helezonik Modelde kullanıcı ara ürünleri gördükleri için sistemin içindedirler. Risk analizi önemlidir. Büyük projeler için uygundur. Hatalar erken giderilir. Planlama vardır fakat çok karışık ve uzun sürdüğü için büyük projelerde kullanılır.

Artımsal geliştirme modelinde müşteri ile içe içe olunması dolayısıyla işlevsellik ön plana çıkar ve proje daha güvenli olur. Gereksinimleri müşteri belirler, özellikler daha fazla test edilmiş olur ancak her artımı tanımlayabilmek için sistemin tanımlanması gerekir. Artımlar kendi içlerinde tekrarlanamaz.

Kodla ve Düzelt modeli küçük projelerde kullanılır. Bakımı zordur ve emeklilik vardır. Fazla maliyetlidir bundan ötürü büyük projelerde tercih edilmez. Program son haline kadar sadece kodlanır ve teslim edilir. Dokümantasyon gibi olaylar yoktur o yüzden kullanımı sadece kişiye özeldir.

Çevik Modellerde müşteri ve ekip iletişimi ve ekip içi iletişim fazla olduğundan hatalar en aza indirgenir. Alışılması basit ve değişime açıktır. Takım çok önemli bir konumdadır. İhtiyaçların sürekli değiştiği için çalışma saatleri uzayabilir. Ekip birbiriyle sürekli iletişim içinde olduğundan günümüzde en çok tercih edilen modeldir.

Evrimsel Geliştirme metodunda kullanıcılar gereksinimleri daha iyi anlar, hatalar azalır. Ancak bakımı zordur.

**Hangi Projelerde Hangi Modeller Kullanılabilir**

Eğer belirsizlik az ise ve iş tanımları belirgin olduğu bilgi teknolojileri projeleri için V Süreç Modeli kullanılabilir.

Eğer hedef kitle büyük ise Evrimsel Geliştirme Modeli kullanılabilir.

Büyük, maliyetli ve uzun süreli bir proje ise Spiral Model veya Artımsal Geliştirme kullanılabilir.

Eğer proje orta veya küçük kitleli ve uzun sürmeyen projelerde Çevik Modeller kullanılır.

Kodla ve Düzelt Modeli ise zaman sorunu olmayan ve küçük projelerde kullanılır.

Eğer proje küçükse ve gereksinimleri iyi belirlenmiş ise Şelale Modeli kullanılır.

**Kaynaklar**

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

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

<https://www.techwell.com/2013/02/why-scrum-so-popular>

<https://furkanalniak.com/yazilim-muhendisligi-yazilim-surec-modelleri/>

Doç. Dr. Deniz Kılınç, Bakırçay Üniversitesi, Yazılım Mühendisliği Temelleri dersi 3. Ve 4. hafta sunumları