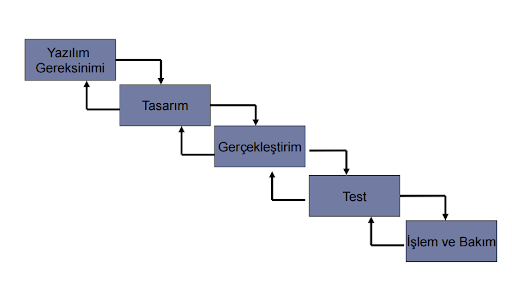
Hızla büyüyen ve gelişen dünyada teknoloji, ve de bu gelişen teknolojide kullanılan yazılımlar gün geçtikçe daha büyük önem kazanmaktadır. Bunun sonucu olarak yazılımların tasarlanması, geliştirilmesi ve servis edilmesi, yani bir yazılımın yaşam döngüsü de oldukça önemli bir gereksinim haline gelmiştir. Bu gereksinimin düzgün karşılanabilmesi için yazılımların yaşam döngüleri için birçok farklı durumda kullanılabilen pek çok farklı modelleme türü ortaya çıkmıştır.

Bu modelleme türlerinden ilki, aslında modelleme dememizin pek de doğru olmadığı “gelişigüzel modelleme”dir. Bu modelleme türü programlamanın yeni bir alan olarak ortaya çıktığı 1960’larda, günümüzdekilerle karşılaştırıldığında çok daha basit olan programların yazımında kullanılmıştır. Herhangı bir planlama bulundurmayan bu modelleme çeşidiyle program yazmak, daha doğrusu yazılan programı okuyup anlamak ve bakımını yapmak oldukça güç olduğundan yerini yeni ve daha kullanışlı modelleme türlerine bırakmıştır. Günümüzde üzerinde tek bir kişinin çalıştığı, oldukça basit programların yazımında hala kullanılmaktadır.

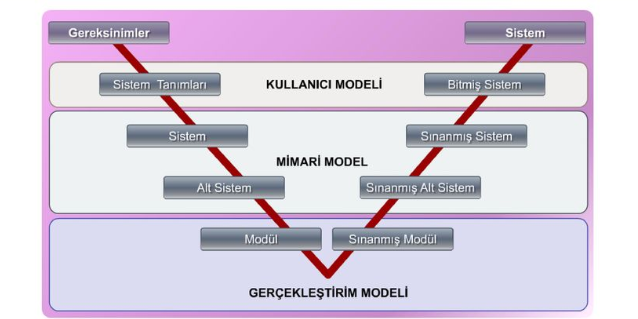
Bir sonraki model, 1970’lerde ortaya çıkmış Barok Modeli’dir. Bu modelde, yazılım-yaşam döngüsündeki bütün adımlar ayrı ayrı ele alınarak sırasıyla gerçekleştirilir. Herhangı bir adımda ortya çıkacak sorunlarla nasıl başa çıkılacağı ise geliştiricelere bırakılmıştır ve kesin değildir. Bu modelleme türünün bir özelliği de dökümantasyon yani belgeleme işleminin, program geliştirildikten sonraya bırakılmasıdır. Bu durum özellikle büyük çaplı projelerde büyük sorun yaratmaktadır. Çünkü projenin hızlıca, üstün körü yapılmış kısımları varsa ya da projenin belli bir bölümünden sorumlu bir kişi bunu nasıl yaptığını anımsamıyorsa bunun telafisi olmayacaktır. Bu nedenlerden ötürü de Barok modeli günümüzde kullanımdan düşmüş, yerini yeni modellere bırakmıştır.



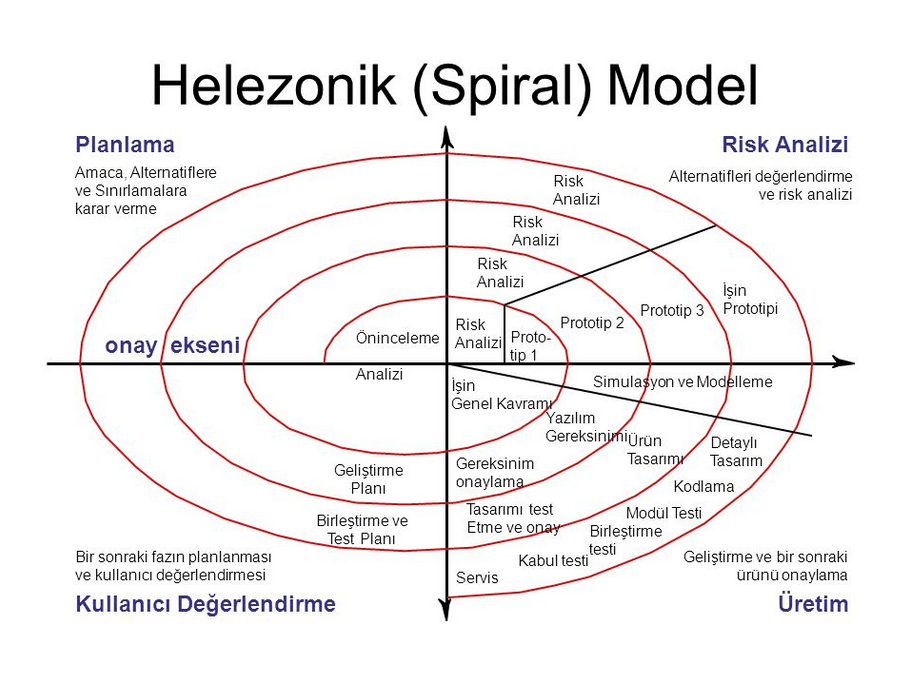
Bu modellerden biri de “Geleneksel Modelleme” olarak da bilinen “Çağlayan Modeli”dir. Bu modelde temel, bütün aşamaların teker teker bitirilip ancak ondan sonra bir sonraki aşamaya geçilmesidir. Barok modelinin tersine Çağlayan modelinde her aşamanın ardından dökümantasyon ve test yapılmalıdır, dökümantasyonu ve testi yapılmamış aşama, tamamlanmamış sayılmaktadır, yani Barok modelinin tersine Çağlayan modelinde projenin dökümantasyonu sürecin bir parçasıdır. Ancak bu modelin bir eksiği, analiz aşamasının olabildiğince ayrıntılı olması ve de bu ayrıntıların bir sonraki aşamalara oldukça iyi yansımasının gerekmesidir. Ki bu da analiz aşamasının projenin önemli bir kısmını oluşturmasına neden olmaktadır. Her ne kadar Barok modelinin tersine Çağlayan’da aşamalar arası geri dönüşlerin nasıl olacağı belli olsa da bu nedenle eksik bilgi ya da yanlış analiz gibi durumlar büyük sonuçlara neden olabilmektedir. Ayrıca aşamalardan sırayla geçileceğinden geri dönüşler maliyetli olmaktadır. Bunun yanında, geçen süre de artacağından programın kullanıcılara ulaşması uzayacak ve iki taraflı bir memnuniyetsizlik oluşacaktır. Bu durum programın maliyetini ve gereken süreyi daha da arttırabilmektedir. Bu model hala kulllanımda olsa da gün geçtikçe daha az kullanılmaktadır.



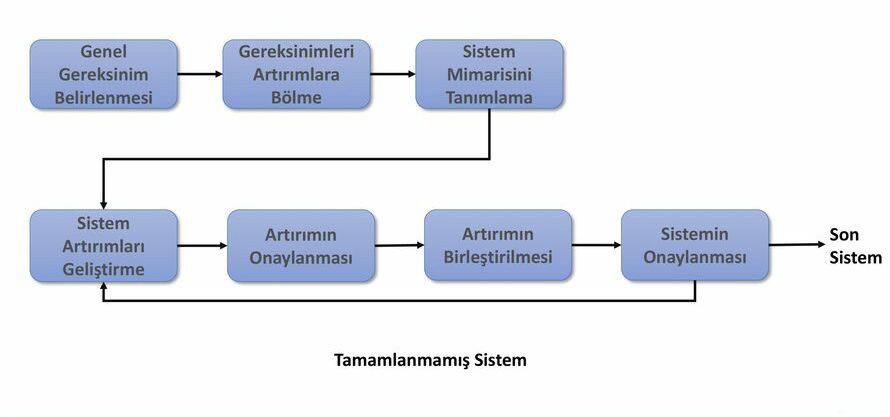
Bir başka modelimiz de “V Süreç Modeli”. Adını çalışma biçiminden alan bu model iki temel aşamadan oluşmaktadır. V’nin ilk kolu olan “üretim” aşaması ve v’nin ikinci kolu olan “sınama” aşaması. Bu aşamalar da kendi içlerinde üç alt modele ayrılmaktadır. Diğer modellemelerde de oldduğu gibi V Süreç Modeli de gereksinimlerin tanımlanmasıyla başlar. Ardından V’nin üretim koluna girilir. Burada kullanılacak ilk model “Kullanıcı Modeli”dir. Bu modelde kullanıcının istek ve gereklerine uygun bir sistemin tasarıları ve planları üretilir. Bir sonraki model “Mimari Model”dir. Bu modeldeyse bir önceki modelde oluşturulan tasarımlar ın sınama işlemleriyle ilgili işlemler yürütülür. Son olarak “Gerçekleştirme Modeli”nde programın yazılması gerçekleşmektedir. Bu model aynı zamanda V’nin ikinci kolunu tırmanmaya başladığı yerdir ve bu nedenle sınama aşamasının başlangıcını da içerir. Bu aşamada sistem test edilip sınanarak kullanıcıya sunulur. V Süreç Modeli’nin bir önceki modellerden bir farkı, kullanıcıyı sürecin bir parçası haline getirmesidir. Örneğin Çağlayan Modeli’nde kullanıcı süreç dışında tutulup yalnızca süreçten önce ya da sonra yazılıma erişebilir, ancak V Süreç Modeli’nde kullanıcıdan dönüt alınmadan süeç bitmiş sayılmaz. Ancak V Süreç Modeli’nde de tıpkı Çağlayan Modeli’nde olduğu gibi, analiz aşamasının ayrıntılı olması gerekmektedir.



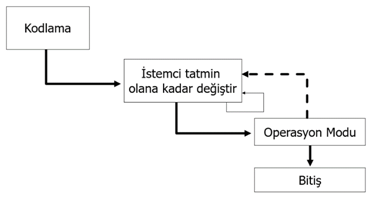
Helozonik ya da Spiral Model sürekli olarak dört fazdan oluşan bir döngü içerisinde bulunan bir yazılım-yaşam modelidir. Bu model de elbette diğerleri gibi bir analizle başlamaktadır. Analiz ardından sırasıyla planlama, risk analizi, üretim ve kullanıcı değerlendirme aşamalarından geçilir. Planlama aşaması adından da anlaşılacağı gibi bir ara ürünün tasarlanmasından oluşur. Risk analizi aşamasında ne tür risklerle karşılaşılabileceği ve bunların nasıl önleneceği kararlaştırılır. Ardından üretime geçilir ve plana göre bir ara ürün oluşturulur. Son olarak bu ara ürün kullanıcı değerlendirmesinden geçerek bu değerlendirmeden alınan dönütlerle yeniden planlama aşamassına geçilir ve süreç böylece devam eder. Bunun pek çok artısı bulunmakadır. İlk olarak, tıpkı V Süreç Modeli’nde olduğu gibi kullanıcı sürecin bir parçasıdır, ancak V Süreç Modeli’nin aksine Helezonik Model’de kullanıcı sürecin son aşaması değildir. Bu durum, kullanıcının istek ve gereksinimlerine ters düşen durumların daha erken fark edilip düzeltilmesine yaramakla kalmaz, ayrıca analiz aşamasına düşen yükü de azaltır. Sonuç olarak Helozonik Model’de sürekli bir geri dönüşlülük bulunmakta ,süreç ve ara ürünün üstüne eklemek kolaylaşmaktadır.



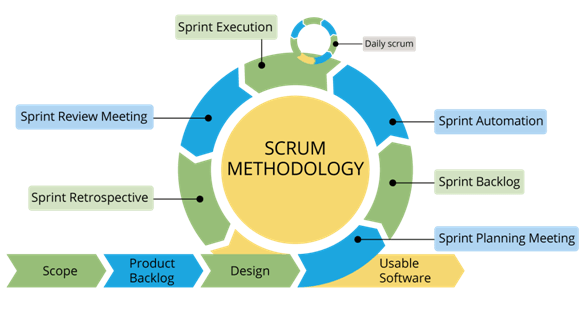
“Arttırımsal Geliştirme Modeli”, bir programın birden çok kere üretilip teslip edilmesi esasına dayanan bir yazılım-yaşam döngü modelidir. Bu modelde programın geresinimleri, bir öncelik sıralamasına konarak bu gereksinimlerden öncelikli olan önce teslim edilerek bir sonraki gereksinim bunun üzerine eklenir ve yeniden teslim işlemi yapılır. Bu sırada ortaya çıkan sorunların giderilmesi işlemi de gene bir sonraki teslime kalır. Bu sayede program bir yandan kullanılırken bir yandan geliştirilmeye devam etmesinin yanında, kullanıcı sürekli olarak programı kullanmakta olduğundan programın işlevsellğini daha erkenden görecektir. Ayrıca program sürekli olarak bir önceki sürümlerin üzerine geliştirildiğinden programın batması riski de büyük ölçüde azalacak ve dolayısıyla projenin maliyeti de risk durumlarından daha az etkilenecektir.



“Kodla Ve Düzel Modelit” bütün yazılım-yaşam döngü modelleri arasından en kolayı ancak aynı zamanda en oahalı olanıdır. Bu modelde daha çok birkaç yüz satırdan oluşan küçük çaplı projelerde, tasarım ya da planlama yapılmadan doğruca kodlama aşamasına geçilir. Hatta projeyle ilgilenen kişi doğrudan müşteri/kullanıcıyla iletişim içerisinde işi sürdürebilir. Bu nedenden ötürü uygulaması kolay bir model olsa da ileriki safhalarda pek çok güçlük çıkarmaktadır. Örneğin bu modelde bir dökümantasyon söz konusu olmadığından kodu yazan kişi dışındakiler kodu okumakta ve gerektiğinde bakım yapmakta büyük güçlükler çekecektir. Öte yandan kolay olması nedeniyle pek çok küçük projede ve de özellikle programlamaya yeni başlamış kişilerin ilk projelerinde yaygın olarak tercih edilmektedir.



Son olarak “Scrum”dan söz etmek istiyorum. Ancak bunun için önce “Çevik Yazılım Geliştirme”nin ne olduğundan söz edeyim. “Çevik Yazılım Geliştirme” günümüzde hızlanan dünyada müşterilerin/kullanıcılıların, gereksinimlerinin sürekli olarak değişmesi ve gereksinimlerin iyi açıklanamaması gibi durumlarla baş etmek adına ortaya çıkmış bir olgudur. Scrum, günümüzde oldukça yaygın bir biçimde kullanılan bir çevik yazılım metodudur. Scrum üç temele dayanmaktadır. Bunlardan ilki “saydamlık”tır, müşteri/kullanıcı olanı biteni bilmek durumundadır. İkincisi “denetleme”dir. Proje düzenli olarak denetlenilmektedir. Üçüncü ve sonuncu temelse “uyarlama”dır. Proje, ileride yapılabilecek değişimlere uyum sağlayabilmelidir. Scrum’da proje bir ile üç hafta arası süren “sprint” adı verilen süreçlere ayrılır ve bu süreçlerin sonunda toplantılar. Yalnız bu toplantılar dışında günlük olarak on-beş dakika kadar bir süre proje çalışanları “Dün ne yaptım, bugün ne yapacağım, çalışmama engel olan bir şey var mı?” sorularını birbirileriyle tartışarak genel bir fikir alışverişinde bulunurlar. Toplantılarda ya da günlük konuşmalarda çalışanların verimini etkileyen herhangi bir iç ya da dış sorunun dile gelmesi durumunda devreye “Scrum Master” adı verilen kişiler girer ve bu sorunu ortadan kaldırırlar. Scrum master ekibin bir parçası olmasına karşın ekibin lideri değildir. Scrum metodunda, projede çalışanlar kendi kendilerini yöneterek herkesin işin bir ucundan tutması mantığıyla çalışırlar. Bunun sonucu olarak aynı işte birden fazla kişi çalışabilmektedir. İş verimini arttırması nedeniyle de günümüzde oldukça yaygınlaşmıştır.



Modelleme çeşitlerinden söz ettiğimize göre şimdi de hangisinin nerede tercih edileceğine değinelim. Öncelik olarak küçük projelerde çok bir beklenti olmayacağından ve de üretim uzun zaman almayacağından Kodla-Düzelt, Gelişigüzel, Barok ya da Çağlayan modelleri kullanılabilir. Burada projenin ne kadar küçük olduğu ve ileride üzerine ekleme ya da çıkarma yapılıp yapılmayacağı önemlidir. Gelişigüzel Model, kişisel bir projede tercih edilebilse de genel anlamda işlevsiz kalacaktır. Kodla-Düzelt ise ileride bakımı zorlaştıracağından yalnızca tek üretim sürecinden oluşan projelerde tercih edilebilir. Barok Modeli, dökümantasyon eksikliği ve aşamalar arası dönüşün belirsizliğinden dolayı aşamaların sırayla gerçekleşeceği projelerde tercih edilecektir. Çağlayan Modelindeyse dökümantasyon ve aşamalar arası geçişler belirli olsa da analiz aşamasının ağırlığı çok olduğundan gene uzun süreli projeler için tercih edilmeyecektir. Kalan modeller arasından V Süreç Modeli, iki aşamada gerçekleşecek projelerde tercih edilir. Helozonik Model, belirsiz aşamadan oluşan projeler için uygundur ancak program geliştirme aşamasındayken kullanımda olamayacağından bu süre içerisinde öncceki sürüme katlanılmak zorundadır. Arttırımsal Geliştirme Modelinde proje hem kullanılıp hem geliştirilebildiğinden özellikle uzun süreli, bir yandan da kullanılan ve sürekli değişlikliğe açık programların geliştirilmesine bir numaralı tercih olacaktır.

Olgun AYDIN 200601026