BAKIRÇAY ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ ESER ELÇİ 180601056

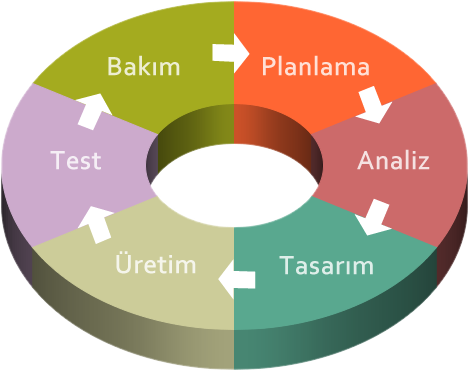
YAZILIM YAŞAM DÖNGÜSÜ (SOFTWARE DEVELOPMENT LİFECYCLE)

Herhangi bir yazılımı geliştirme işi sadece kodlamalara dayanmamaktadır. Yazılım geliştirme işi birden fazla adımdan oluşmaktadır. Kodlama, bu adımlardan sadece bir tanesidir.

Bir yazılım projesi geliştirilirken planlama, analiz, tasarım, üretim ve test aşamaları yer almaktadır. Bir yazılım projesi; yazım aşamasında, geliştirilme sürecinde, müşterideki kullanım süresi boyunca ve bakım zamanında geçirmiş olduğu tüm adımlara kısaca projenin planlamasından müşteriye teslim ve müşterinin kullanım süresi içinde geçirmiş olduğu tüm aşamalara yazılım geliştirme yaşam döngüsü adı verilir. Bu aşamalar bir döngü halinde gerçekleşmektedir.

Bu aşamalar, yazılım fonksiyonları ve gereksinimleri sürekli olarak değiştiği ve geliştiği dolayısıyla bir döngü şeklinde düşünülmelidir. Yazılım projesi tamamlandıktan sonra müşteriden gelecek istekler ve bu doğrultuda projeye eklenecek yeni özellikler, modüller vs. için yazılım geliştirme yaşam döngüsünde geriye dönmek veya ileriye gitmek mümkündür.

Herhangi bir yazılım projesinde planlama, analiz, tasarım, üretim ve test aşamalarının bittiğini ve bu aşamalar tamamlandıktan sonra hiçbir hata ve yeni bir talep olmadığını varsayarsak bu sefer de projenin bakım aşaması devreye girmektedir. Geliştirilen veya geliştirilecek olan bütün yazılım projelerinin sonraki zamanlarda ortaya çıkabilecek hataları ve müşteriden gelen talepleri için de bakım süreci yazılım yaşam döngüsüne eklenmektedir.



1) Planlama: Yazılım geliştirme projesinin ilk aşamasıdır. Müşteriden gelen talepler doğrultusunda işin projelendirilip bir fikrin ortaya çıkmasıyla beraber fikrin belirginleştirilmeye çalışıldığı ve bu fikrin tartışıldığı aşamadır. Bu aşama yazılım geliştirme projesinin ne kadar zaman içerisinde, nasıl bir maliyetin gerektiğinin hesaplanması ile planlanan ve projenin hayata nasıl geçirilebileceğinin düşünüldüğü evredir.

2) Analiz: Planlamadan sonra gerçekleştirilen planlanmış olan projenin tanımlarının yapıldığı aşamadır. Bu aşamada sistem veya yazılım projesini hayata geçirebilmek için temel işlevlerin ve gereksinimlerin belirlenip bunların belgelendirilmesidir. İstenilen projedeki gereksinimlerin tam olarak anlaşılmadığı durumlarda müşteri ile iletişime geçilir ve karşılıklı iletişime dayalı bir şekilde anlaşılmazlık en düşük seviyede tutulmaya çalışılır.

3) Dizayn: Planlama ve analiz aşamasına göre tasarım çizilmeye başlanır. Bu aşamada kararlar verilir ve tercihler yapılır. Örnek olarak yazılım projesinde ekran neler bulunacağı, ekrandan ekrana nasıl geçileceği, yazılım bileşenleri ve modülleri bu aşamada tasarlanır. Bir sonraki adıma bütün kararların ve tercihlerin yapılarak geçilmesi beklenir. Bu aşamada herhangi bir belirsizlik veya soru bırakılmaz.

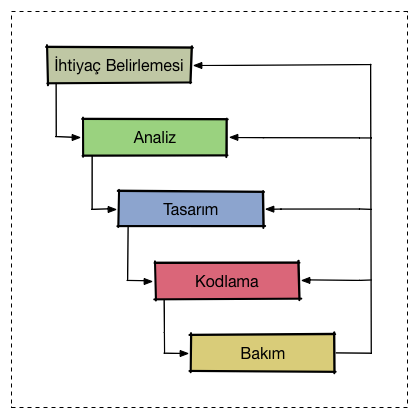
4) Üretim: Tasarım aşamasının bitmesiyle beraber yazılım projesinin kodlanmasına başlanır. Bu aşamada yazılım projesinin gelecekteki bakımlarının daha pratik olması için, bu projenin yeni mezun olan birisinin bile anlayabileceği kadar karmaşık olmayan, belirli kodlama standartlarına sahip bir şekilde kodlamanın başlaması gerekir.

5) Test: Bu aşamada yazılım projesi veya sistem artık gerçek hayata geçirilmiştir. Hayata geçirilen bu proje bir test ekibi aracılığıyla beta testleri yapılarak herhangi bir hata çıkmadıysa müşteriye sunulur. Hata çıkar ise döngüde geriye dönülür ve bu hata giderilmeye çalışılır. Ardından yazılım projesi önceden hazırlanan kullanım kılavuzu ile müşteriye teslim edilir.

6) Bakım: Bu aşama yazılım projesinin veya sisteminin müşteriye teslimi ile başlar. Yazılım projesi veya sistemi müşterinin kullanım süresi içerisinde vermiş olduğu hataların düzeltildiği, ek güncellemeler ile yazılımın iyileştirilmesinin amaçlandığı ve yeni geliştirmelerinin eklendiği aşamadır.

Yazılım Geliştirme Modelleri

1)Şelale (Waterfall) Modeli



Çok iyi bir şekilde tanımlanan dokümantasyon ağırlıklı, kolay bir şekilde anlaşılabilen ve üretimi kısa süren basit yazılımlar için uygundur. Bu yöntemde yazılım geliştirme süreci ihtiyaç belirlenmesi, analiz, tasarım, kodlama ve test, bakım gibi aşamalardan oluşur. Bu modelde bir adım bitirilmeden diğer bir adıma geçilmez.

Avantajları

\*Parçalara bölünmüştür.

\*İhtiyaç belirlendikten sonra temeller oluşturulur.

\*Bu yöntemle projenin yapım aşaması kısa sürer.

\*Kullanımı ve anlaması kolaydır.

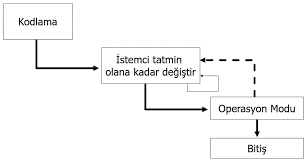
Dezavantajları

\*Uzun ve anlaşılması zor projelerde istenilen performansı verememektedir.

\*Projede çıkan hataların düzetilmesi için bazen birden fazla kez geriye dönmek gerekmektedir ve bu da maliyetleri arttırmaktadır.

\*Bir aşama bitmeden diğerine geçilemez. Aksi halde eksiklikler ve hatalar ortaya çıkar.

2)Kodla-Düzelt Modeli



Bu model çoğunlukla resmi olmayan projelerde kullanılır. Çünkü bu modelde herhangi bir yöntem, dokümantasyon bulunmamaktadır. Kodlamayla başlar ve ürün hazır oluncaya kadar da kodlama ile biter.

Avantajları

\*Programdaki aşamalar hızlı bir şekilde geçilir.

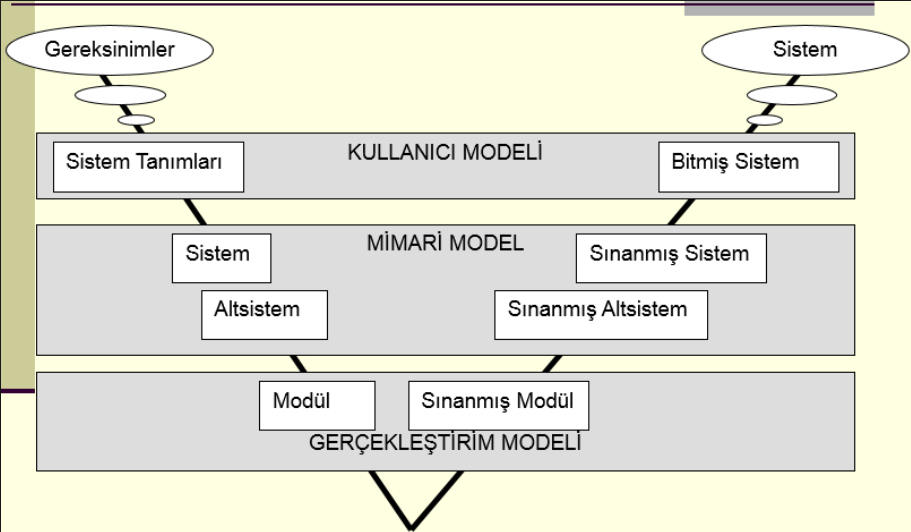
Dezavantajları

\*Herhangi bir plan gerekmez.

\*Bitiş süresi istemciye bağlı olarak değişir. Kesin bir zaman söylenmez.

\*Kodlar başlangıç seviyesiyle yazıldığı için kodları düzeltmek maliyetlidir.

3)V Modeli



Bu modelin kullanılabileceği projelerde belirsizlik az ve iş tanımlarının belirgin olması gerekmektedir. Şelale modelinin biraz daha geliştirilmiş halidir. Doğrusal bir yönde ilerlememektedir. Kodlama evresinden sonra yukarıya doğru bir eğim alır ve V şeklini oluşturur. Bundan dolayı ismi V modeli olarak anılmıştır. Test odaklı bir modeldir. Testler dokümantasyon işlemiyle başlar.

Avantajları

\*Kullanım açısından basittir.

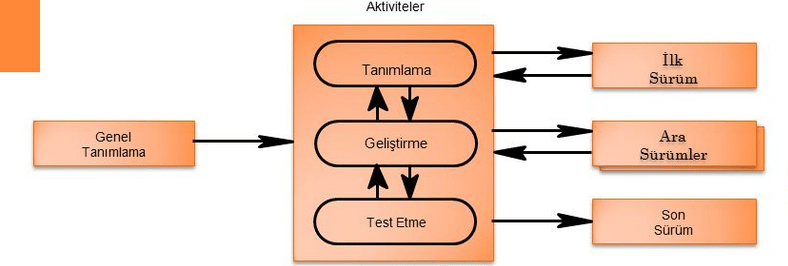
\*Projenin takibini kolaylaştırmaktadır.

\*İleriye yönelik program yazılmasını sağlar.

Dezavantajları

\*Maddeler arasında yinelemeler kullanılmaz.

4)Evrimsel Geliştirme Modeli



Coğrafik olarak geniş bir alana yayılmış olan birden fazla birimi olan kuruluşlarda kullanılması tavsiye edilen bir modeldir. Diğer modeller ile kıyaslandığında adımlar daha yavaş ilerlemektedir. Modelin genel başarısı ilk aşamanın başarısına bağlıdır.

İki çeşit evrimsel geliştirme modeli vardır:

\*Keşifçi Geliştirme

Amaç: Müşteri ile karşılıklı iletişim halinde olarak müşterinin gereksinimleri inceleyip ve beraber çalışıp sistemi veya yazılım projesini teslim etmektir.

Keşifçi geliştirme için anlaşılır bir gereksinim ile başlamak lazımdır.

\*Atılacak Prototipleme

Amaç: Yazılım projesi veya sistem için gereksinimlerini anlamak.

Net bir şekilde anlaşılmamış olan gereksinimlerle süreç başlar.

Avantajları

\*Gereksinimlerin daha iyi anlanmasını sağlar.

\*Hemen hemen her aşamadaki ileriye ve geriye dönüşler hata riskini azaltır.

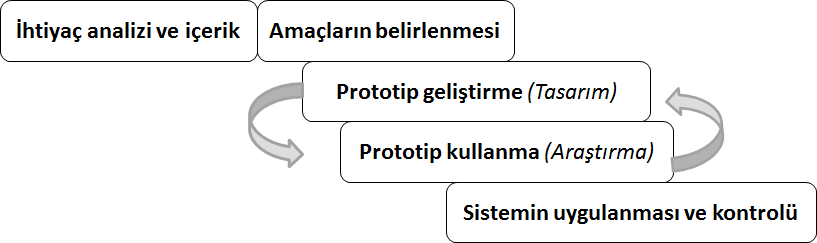
Dezavantajları

\*Sistemin veya yazılım projesinin bakım yapılabilirliği zordur.

\*Sistemin veya yazılım projesinin gereksinimlerini birçok kez değiştirmek gerekebilmektedir.

\*Gereksinimlerin birçok kez değiştirilmesiyle birlikte yazılım projesi veya sistem iyi bir şekilde yapılandırılamaz.

5)Prototipleme Modeli



Prototipleme modelinde isminden de anlaşılacağı gibi amaç bir prototip ortaya çıkarmaktır ve bu prototipi geliştirebilmektir. Bu modelde hızlı plan, tasarım, kodlama ön plana çıkmaktadır. Kullanıcılar ile geliştiriciler bir masa etrafında istenilen yazılım projesi veya sisteminden çıkarılabilecek bütün çıktılara ve bu çıktıların oluşması için hangi girdilerin olması, hangi işlemlerin gerçekleşmesi gerektiğine karar verirler. Ardından tez bir şekilde yapılan bir tasarım ile yazılım kullanıcıya yansıyacak yönünü aktaran ilk prototip üretilir. Bu prototip kullanıcıya sunulur. Kullanıcı bu prototipi kullanır ve değerlendirir. Bu değerlendirme sonuçlarına göre prototip üzerinde değişiklikler, geliştirmeler yapılarak kullanıcının isteği karşılanmaya çalışılır.

Avantajları

\*Kullanıcı ilk prototip sayesinde eksik söylediği gereksinimleri görebilir.

\*Kullanıcıdan gelecek olan yeni gereksinimlere açıktır.

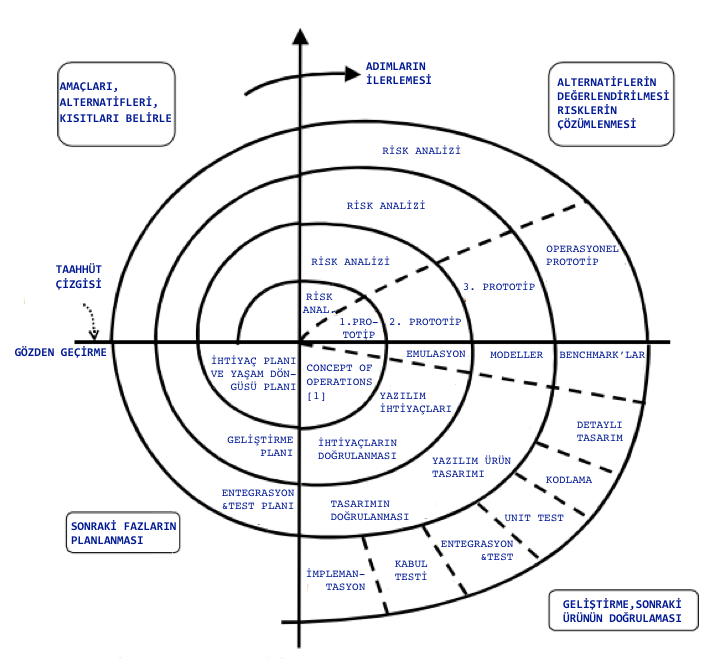
\*Karşılıklı iletişim sorunu olan yanlış anlaşılmaları engeller.

Dezavantajları

\*İlk prototipin belgelendirmesi eksik veya yoktur.

\*Prototipin düzeltme aşaması geçilirse düşük performanslı yazılım projesi veya sistemi oluşur.

6)Spiral Model



Spiral modelde prototipleme ve risk analizi en temel konulardır. Şelale modelinde görmezden gelinen riskleri her döngü öncesinde analizini yaptırarak hata riskini düşürür. Yazılım projesi veya sistem devirlere ayrılır ve devirlerin her birinin risk analizi yapılır. Bu modeli diğer modellerden ayıran en önemli özellik risk analizinin detaylı bir şekilde yapılmasıdır. Planlama, risk analizi, üretim, kullanıcı değerlendirmesi gibi devirlerden oluşur.

Avantajları

\*Yazılım projesinin veya sistemini küçük parçalara ayırır. Bu sayede risk payının fazla olduğu kısımlar ilk olarak ele alınır.

\*Çözüm için döngü küçük parçalara ayırıldığı için bölümlerdeki hatalar erken tespit edilip giderilmeye odaklanılır.

\*Birçok yazılım modelini içinde bulundurur. Bundan dolayı en çağdaş modellerden biridir.

Dezavantajları

\*Küçük ve risksiz yazılım projeleri veya sistemleri için maliyetlidir.

\*Diğer modellere göre karmaşık bir yapıya sahiptir.

\*Belgeleme işlemleri fazladır.

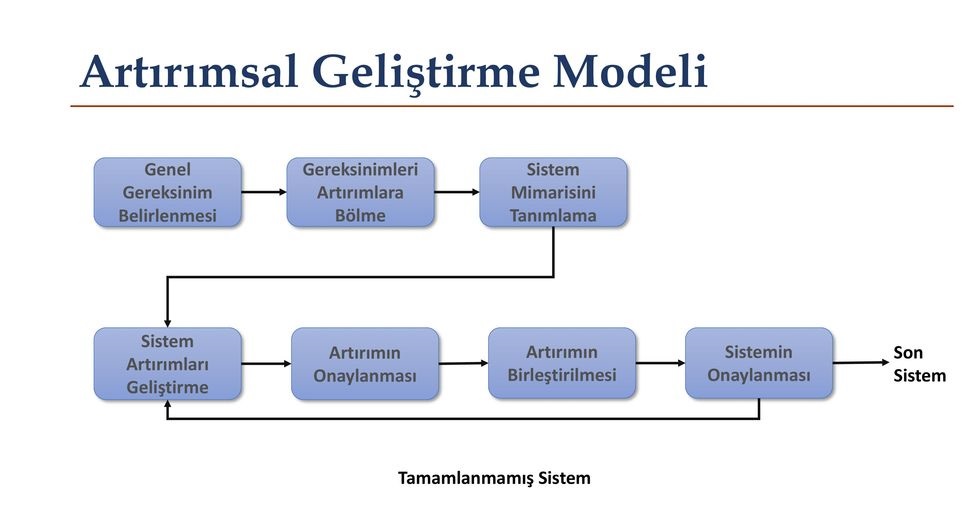
7)Gelişigüzel Model

Bu yöntemi tam bir model olarak adlandıramayız. Çünkü herhangi bir yöntem veya model içermez. Takip edilebilirliği, izlenebilirliği, bakım yapılabilirliği, okunurluğu zor ve düşük olan bir modeldir. Çoğunlukla kişiye bağlı olarak geliştirilen yazılımlarda kullanılır. Çoğunlukla tek bir kişinin kullandığı yöntemdir.

8)Barok Modeli

Bu model 70’li yıllarda kullanılan adı gibi eski bir modeldir. Bu modelde belgeleme ayrı bir işlem olarak ele alınır. Ancak günümüzde belgeleme işlemi normal bir işlem olarak görülür. Aşamalar arasında geri dönüşlerin nasıl yapılacağına dair herhangi bir tanım bulunmamaktadır.

9)Artımsal Geliştirme Süreç Modeli



Bu modelde yazılım projesi üretilirken ana yapı korunarak proje geliştirilir. Müşterinin gereksinimleri alınır ve bu gereksinimler önem sırasına sokulur. En önemli olan gereksinimler projede erken teslim ile müşteriye teslim edilir. Proje tek seferde müşteriye teslim edilemez müşterinin taleplerinin öncelikliği doğrultusunda parçalara ayrılarak teslimat yapılır.

Scrum (Agile Yöntemi) Günümüzde Neden Popülerdir?

Yüksek verimi, kısa döngülü bir şekilde çıktı ve geri bildirim üretebilmeyi hedefleyen günümüz yazılım geliştirme ortamlarına için dizayn edilmiş bir yöntemdir. Yazılım projesi hayatı boyunca önemlilik sırasına göre gereksinimleri sıralanır ve bu sıralamaya göre de geliştirilir. Scrum’un sektöre kazandırdığı en önemli fayda ise yazılım projesinin ilerlemesinin devamlı bir şekilde takip edilebilmesi ve yapılması planlanan özellikler / geçmiş olan gün sayısı grafiğidir. Çıkarılan grafik sayesinde yazılım projesini yönetenler projenin ne durumda olduğunu istediği an görebilmektedir. Önem sırasına göre önceliklendirilen gereksinimler küçük yapılara ayrılarak daha küçük alt projeler oluşturulur. Proje küçük parçalara ayrıldığı için artık projeyi geliştirmek daha kolay olmuştur. Güzel bir planlamayla birlikte birkaç adımda ortaya muhtemel bir ürün çıkar. Bu ürün sayesinde müşteri ile beklediği gereksinimlerin karşılanıp karşılanmadığı ile alakalı sonradan çıkması muhtemel problemleri karşılıklı olarak erkenden tespit etmemize olanak sağlar. Yani şelale modelinde bir kez tekrarlanan döngü Scrum’da defalarca tekrarlanır. Bu şekilde Scrum’da yazılım ekibinin ve yapılacak olan yazılımın kullanıcılarının belirli bir süre içerisinde en yüksek verimle çalışması tüm yazılımın ise bir aylık döngülerden oluşan daha büyük bir proje sonucunda üretilmesini hedefler. Bu sebeplerden dolayı da Scrum günümüzde popülerdir. Scrum’un kötü olan tarafları ise maliyetlerin yazılımda sürekli bir değişim olabileceğinden dolayı tahmin edilememesi, müşteri ile iletişime geçildiği takdirde müşterinin negatif konuşmalarının sizi alaşağı etmesidir.