

DANIŞMAN ÖĞRETMEN: Deniz KILINÇ

Adı : Çağlar

Soyadı : HEKİMCİ

Bölümü : Bilgisayar Mühendisliği

Numarası : 180601040

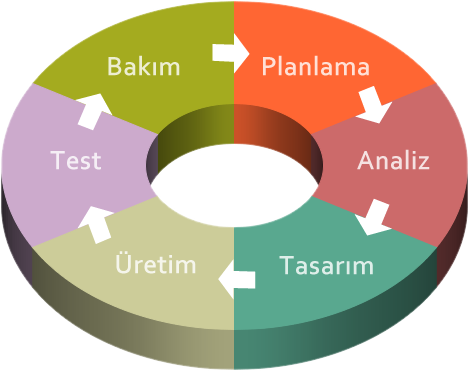
YAZILIM YAŞAM DÖNGÜ MODELLERİ HAKKINDAKİ MAKALEM

YAZILIM NEDİR?

Bilgisayarın fiziksel ürünlerini kullanan, çeşitli görevleri yapmak için bilgisayarın anlayabileceği yani makine dili denilen dilde tasarlanan komutlara yazılım denir. Yazılım; bilgisayarın fiziksel ürünleri olan donanımını kullandığı gibi bu dijital parçaların kendi aralarında yaptıkları elektronik iletişimi de sağlar. Makine dili bilgisayarın anlayabileceği tek dildir. Sadece 0 ve 1’lerden oluşan ikili işlemleri yapabilen bu dil 1. ve 2. kuşak programlama dillerine baktığımızda her bilgisayarın kendine özgü Makine Bağımlı olurken ilerleyen kuşaklarda bu Makine Bağımsız olarak karşımıza çıkmaktadır. 1956 senesinde IBM tarafından geliştirilen Fortran dili bundan 10 sene sonra 1966 senesinde "American National Standards Institute" tarafından resmi olarak ilk programlama dili olmuştur. Fortran dli Makine Bağımsız bir dildir yani her türlü bilgisayarda kullanılabilir. Birçok matematik formülü içeren programların yazımında ve mühendislik uygulamalarında halen daha tercih edilir. 1959 yılında FORTRAN’ın tüm işlevlerini içermekle birlikte giriş/çıkış(1/0) özellikleri eklenerek COBOL geliştirilmiştir. İkinci en yaygın dil olmasında veri saklama, verilerin alınma özellikleri ve raporlamadaki kolaylıklar COBOL'u yaygınlaştırmıştır. 1964 yılında COBOL ve FORTRAN dilleri çok zor olduğu için “BASIC” dili ortaya çıkmıştır. 1970'lerin başında bütünleştirici (sembolik makine dili yani Assembler) dil niteliğinde fakat makineden bağımsız dil açığı yüzünden önce "B" olmak kaydıyla "C" dili geliştirilmiştir ve buna iki buçuğuncu kuşak dili denmiştir. [Yazılım yüzünden meydana gelen en büyük hata, ABD’de Venüs gezegenini keşfetmesi için üretilen “Marier 1” isimli uzay aracının yörüngesinin dışına çıkması olmuştur. 22 Temmuz 1962 sabahı uzaya fırlatılan araç 4 dakika sonra belirlenen yörüngenin dışına çıkar ve herhangi bir yere zarar vermemesi için imha edilir. Yapılan araştırmalar, FORTRAN dilinde yazılan programda aynı eşitlikte kullanılan yanlış bir karakterin buna yol açtığı ortaya çıkar.  Bu da yazılımın ne kadar önemli olduğunu ve ne kadar hassas olduğuna dikkat edilmesi gerektiğini bir kez daha gözler önüne seriyor.]**1**  **İyi Yazılım;** değişen ihtiyaçlara cevap verebilecek nitelikte olmalıdır. Bunun için gerekirse sonrasında güncelleme veya bakım yapılabilmelidir. Başka bir yazılımcı tarafından okunduğunda ise kolayca anlaşılabilecek şekilde olmalıdır. Yazılım dili veya çalıştığı sistem gözetmeksizin, müşterinin sorununa çözüm olan her yazılım iyidir.

Yazılım Geliştirme Yaşam Döngüsü (SDLC)

Bir yazılım ortaya koymak sadece kodlama yapmaktan ibaret olmamakla birlikte içinde birden çok adım bulundurmaktadır. Ana hatlarıyla bu adımlar Planlama, Analiz, Tasarım, Üretim ve Test aşamalarıdır. Bu adımlar bir kez gerçekleştirildikten sonra yazılım bitmiş sayılmaz. Bu yüzden bunlara adımlar denmektense Yazılım Geliştirme Yaşam Döngüsü demek daha uygun olur. Çünkü yazılım ile ilgili gereksinimler sürekli olarak değiştiği için söz konusu aşamalar sürekli bir döngü biçiminde ele alınır. Döngü içerisinde herhangi bir aşamada geriye dönmek ve tekrar ilerlemek söz konusudur. Yazılım yaşam döngüsü tek yönlü ve doğrusal değildir.

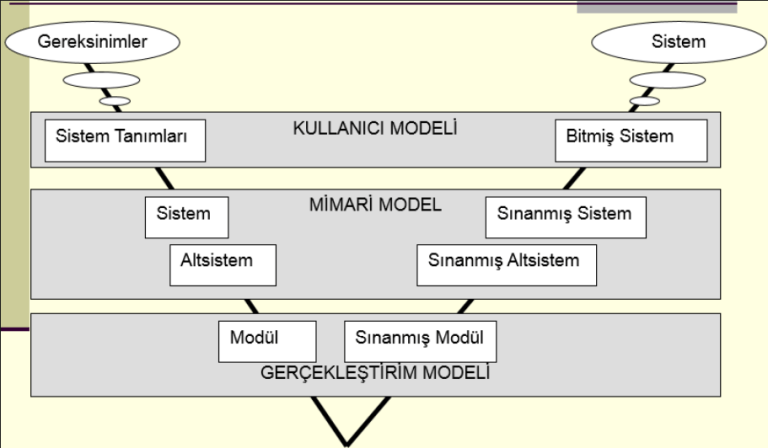
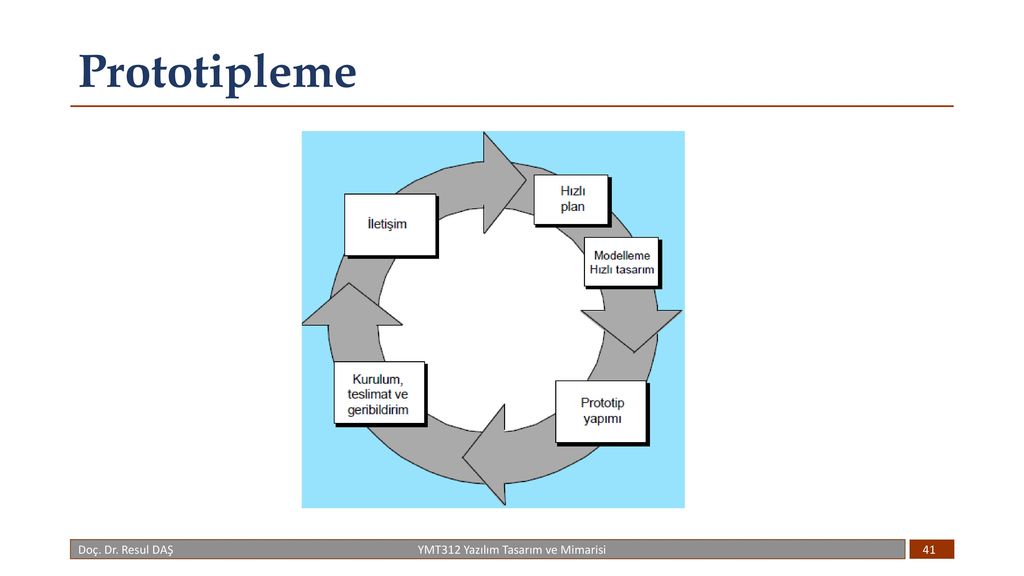
**2**

1. **Planlama:** Bir projeye başlangıç aşamasıdır. Müşterinin gereksinimleri incelenir. Soruna çözüm yolunun ne olacağı birden çok sonuca varılarak ortaya konulur. Proje hakkında konuşulup kesin çözümün kararlaştırıldığı, döngünün en mühim adımlarından biridir.
2. **ANALIZ(TANIMLAMA):** Projenin yazılımsal veya sistem gereksinimlerinin ayrıntılı olarak tanımlandığı aşamadır. Projenin ilerleyen adımlarında ne olacağı ve gidişat hakkında bilgiler içerir. Yazılımın ne yapmak istenildiği kararlaştırılır.
3. **TASARIM:** Bu aşamadan sonra soru işareti kalmayacak şekilde diğer aşamaya geçilinir. Daha çok görüntü üzerine yoğunlaşılır. Bütün efektler, görseller, ara yüz planı netleşir ve ortaya zihinde canlanabilecek bir program taslağı konur.
4. **ÜRETİM(GERÇEKLEŞTİRME):** Herkesçe bilinen Kodlama, Test, Kurulum aşamaları bu adımda hayata geçirilir. Kod yazılmaya başlanılır. Yazılan kod olabildiğince temiz ve kolay anlaşılabilir olmalıdır ki döngü başa sardığında tekrardan bu adıma gelindiğinde ne yapıldığı kolayca anlaşılabilsin. Sonuç olarak proje tamamen hayata geçirilmiş bulunur. Artık böyle bir yazılımın varlığından konuşulabilir.
5. **TEST:** Proje artık hayata geçmiş bulunmakta ve yazılımın birçok farklı kullanıcı tarafından deneme aşamasına girmiş bulunmaktadır. Artık proje sahibinin projeyi kullanacak donanımları temin etmesi, fiziksel şartları ortaya koyması gerekmektedir. Neredeyse projenin bittiği, yazılımın hazır olduğu aşamadır. Tabiki anlatılan olay bir döngüden ibaret olduğu için her an geri adımlara gidilip baştan alınılabilir.
6. **BAKIM:** Test aşamasından doğan sorunları çözmeye, geri dönütleri değerlendirip yazılımı iyileştirmeye yarayan adımdır. Bu şekilde yaşayışına devam eden proje haliyle bu döngüyü de diri tutarak her seferinde yaşanabilecek sorunlara veya doğabilecek yeniliklere cevap verebilecek nitelikte olmalıdır.

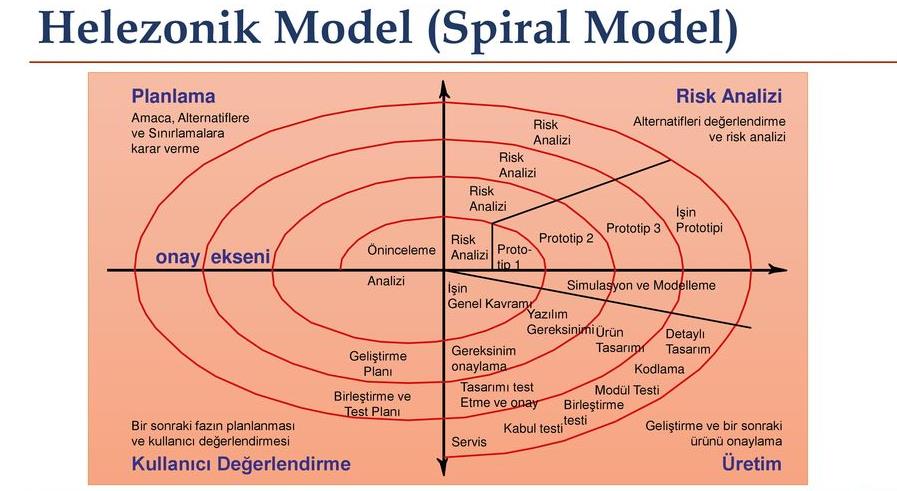
Bu çekirdek süreç denilen adımların dışında Belgeleme ve Emeklilik gibi birçok adım da eklenebilir. Belgeleme geçmişte Barok Modeli örneğindeki gibi ek bir adım olarak görülürken günümüz modellemelerinde ek adım olarak görülmemektedir. Belgeleme denen aşama her adıma ek olarak döngüye eklenebilir. Her geçilen adımda ne yapıldığı ne yapılmadığı, neyin planlanıp neyin ortaya konulduğu hakkındaki tüm bilgiler not edilmelidir. Bu sayede işleyen proje çok daha rahat gözlemlenebilir ve yanlış giden bir şeyler varsa bile buna karşı konulabilir. Emeklilik aşaması ise artık müşterinin sorunu geçtiği, haliyle de yazılımın yani projenin canlılığını yitirdiği ve ihtiyacın kalmadığı durumlarda döngüyü bitiren aşamadır. Bundan sonra yazılım ve proje dosyaları kaldırılıp rafa konulur.

Yazılım üretme projesinde rehber niteliğinde bize yol gösteren modeller şu şekildedir;

1. **Gelişigüzel Model**
2. **Kodla ve Düzelt**
3. **Barok Modeli**
4. **Çağlayan (Şelale) Modeli**
5. **V Modeli**
6. **Helezonik (Spiral) Model**
7. **Evrimsel Model**
8. **Artırımsal Model**
9. **Araştırma Tabanlı Model**
10. **Gelişigüzel Model:** Herhangi izlenecek yöntem yoktur. Hatta ve hatta ortada model söz konusu değildir. Haliyle takibi de bi o kadar zordur. Bu model geliştiren kişiye bağlı olduğundan genellikle tek kişilik ekip ortamında kullanılır.
11. **Kodla ve Düzelt:** Anlaması biraz zor bir modellemedir. Çünkü başlangıç harici eklentilere veya değişen sorunlara çözümü her seferinde üzerine eklenerek çözüme ulaşılmaktadır. Bu sayede yazılım gittikçe karmaşık bir hale dönüşür. Karmaşıklaşan yazılımda regresyon hatası(bir programın herhangi bir bölümünde yapılan değişikliğin başka bir bölümde hata ortaya çıkarması) görülme riski vardır. Yazılım ortaya konulurken takım çalışmasına ortam hazırlamaz. Bireysel kullanıma veya yazılımı ortaya koyan geliştiriciler tarafından kullanılmayacaksa çok önerilmeyen bir modeldir.
12. **Barok Modeli:** Günümüzde popülerliğini kaybeden bir model olmuştur. Bunun en temel nedeni yaşam döngüsünün tam zıttı olarak çalışması yani tek yönlü ve doğrusal olmasıdır. Adımlar bellidir ve döngü şeklinde geri dönüşler tanımlı değildir. Güncel modellemelerde belgeleme bir adım olarak görülmekten çok gayet doğal bir şekilde her aşamaya ek olarak görülmektedir. Fakat Belgeleme adımı Barok Model’inde ayrı bir adım olarak görülmektedir.
13. **Çağlayan (Şelale) Modeli:** Yazılım Yaşam Döngüsü merkezli bir modellemedir. Müşteri ortaya konulacak olan yazılımı biraz daha geç görür. Zamanla müşteri oluşabilecek mali zararı düşünür ve güven kaybı yaşar bu da müşteriyi tedirgin eder. Ama bunun yanı sıra müşteri gelişen projeye birebir tanıklık edebilir ve anlayabilir. Uzun sürmeyecek projelerde kullanılan bir projedir fakat yaygınlığı zamanla azalmaktadır. Bir aşama biter diğer aşama başlar ve tekrarlamalar çok sık görülür. Bu da maliyeti arttırır. Bu yüzden gereksinimler çok iyi tanımlanmış, net olmalıdır.
14. **V Modeli:** V’nin kolları üretim ve sınama bölümlerini temsil etmektedir. Model yatay olarak 3 kısma ayrılmıştır; Kullanıcı Modeli, Mimari Model, Gerçekleştirim Modeli. Kullanımı ve müşteri tarafından takibi kolaydır. Risk en aza indirilmiş durumdadır. Müşteriye uygun tasarım bölümüne Prototipleme denir. Prototipleme gereksinim tanımlama kısmında yapılır.

**3**   **3**

1. **Helezonik (Spiral) Model:** Planlama, Risk Analizi, Üretim, Kullanıcı Değerlendirmesi şeklinde 4 başlıktan oluşmaktadır. Projenin hayata geçeceği bu süre zarfında riskler ön planda tutulur ve çözüme açıktırlar. Aslında birden çok modelin iç içe geçmesiyle oluşan bir modellemedir. Peş peşe çok fazla adım olması çok fazla belgeleme yapılacağı anlamına gelir fakat aslında bu bir yerde avantajdır da. Her ne kadar büyük projeler için birebir olsa da küçük projeler için de pahalı bir modellemedir.



1. **Evrimsel Model:** İlk tam ölçekli model diye geçen bu model sürümün üstüne bir başka üst sürüm piyasaya çıkararak ilerleyen modellemedir. Çift taraflı çalışmak mümkündür, bir taraf kullanırken diğer taraf üretime devam edebilir. Bu sayede uzun zaman alabilecek projelere uygundur. Fakat her cihazdaki yazılım bir üstü çıktığında güncel tutulmalı ve bunun takibi aynı zamanda denetimi yapılmalıdır. Çok büyük alanlara yayılması gereken programlarda daha çok tercih edildiği için değerlendirmesi de birçok kullanıcıya ulaşmakta ve geri dönütlerle birlikte hata sayısı en az ve müşteriyi en fazla memnun eden modellemelerden birisidir. Bunca güzelliklere rağmen bakımı zordur.

1. **Artırımsal Model:** Bu model isminden de anlaşılabileceği gibi yeni çıkan sürümün eski sürümlerin birleştirilip daha çok işlem içeren türe dönüştürülmesiyle oluşturuluyor. Gereksinimler önceliklidir. Sistem tek seferde teslim edilmez. Bütün bir projenin tamamen başarısız olma riski azdır çünkü hayata geçirirken adım adım ilerleyen bir modeldir.
2. **Araştırma Tabanlı Model:** En yaygın bilinen diğer bir deyimi de Yap At Prototipidir. Araştırma ortamları belirsizlik içindedir. Haliyle ortaya çıkabilecek sonuçlar belirgin değildir.

Günümüzdeki metodojilerden (yazılım yaşam döngüsü aşamaları boyunca kullanılacak ve birbirleriyle uyumlu yöntemler bütününe denir) en yaygınları Çağlayan veya Helezonik modeli temel alınmaktadır. En yaygın metodolojiler; SCRUM, Extreme Programming (XP), FDD, MSF…

**XP:** Grup içi iletişimine önem verir. Geri dönüşlere imkân sağlar. İletişim (yüz yüze olmalıdır), Geri Bildirim, Basitlik, Cesaret denilen bu 4lüyü esas almıştır.

**Çiftli Programlama (Pair Programming):** Usta-Çırak ilişkisini ele alıyor. Yeni başlayanlar için çok etkili bir öğrenme şeklidir.

**SCRUM:** Bir işi küçük parçalara ayırarak, çok uzun sürmemekle birlikte günlük toplantılarla iş takibi yapılmaktadır. Üç temel kavramı vardır; Roller, Toplantılar, Araçlar/Bileşenler. Kısa vadelidir.

**KAYNAKÇA**

* **1 =** *https://www.uzaktanegitim.com/haberler/yazilimin-tarihcesi/261*
* **2 =** *http://buluton.com/kurumsal/nasil-calisiyoruz/yazilim-yasam-dongusu*/
* **3 =** [*https://slideplayer.biz.tr/slide/12386328/*](https://slideplayer.biz.tr/slide/12386328/)
* *http://www.ugurkizmaz.com/YazilimMakale-1825-Yazilim-Yasam-Dongusu-Nedir.aspx*
* *http://ybsansiklopedi.com/wp-content/uploads/2015/08/Yaz%C4%B1l%C4%B1m-Geli%C5%9Ftirme-Modelleri-Yaz%C4%B1l%C4%B1m-Ya%C5%9Fam-D%C3%B6ng%C3%BCs%C3%BCSDLCYBS.pdf*
* *https://www.codex.com.tr/yazilim-gelistirme-modelleri*