YAZILIM YAŞAM DÖNGÜ MODELLERİNİN

KULLANIM ŞEKİLLERİ VE İŞLEVLERİ

DELİL TEMEL,

210601004,

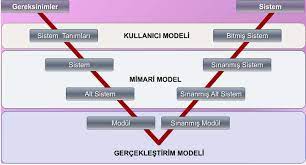
(BİLGİSAYAR MÜHENDİSLİĞİ, BAKIRÇAY ÜNİVERSİTESİ, İZMİR)

GİRİŞ:

Yazılım tıpkı yaşayan bir varlık gibi doğar, gelişir ve zamanı gelince yok olur. Yazılımın gerçekleştirmiş olduğu bu yaşam çevrimine ‘yazılım yaşam döngüsü’ adı verilir. Bu döngü standart bir yol haritası izleyen veya tek yönlü bir sistem değildir. Belirli temel aşamalardan oluşan, dinamik ve esnek bir yapıdadır. Temel olarak şu adımlardan oluşur: Planlama, analiz, tasarım, gerçekleştirim, test, bakım. 1.0) **PLANLAMA:** Projenin planlamasının ve görev dağılımının yapıldığı evredir. Kısa, uzun veya orta vadeli olarak projeye hazırlanılır. 1.1) **ANALİZ**: Projeye başlamadan önce yapılan görüşmelerden sonra planlanan projenin tanımlarını yapıldığı aşamaya geçilir. Projede istenen fikrin veya temel amacın tanımlandığı evredir. Temel problem veya yazılımın yaşam döngüsünün tanımlandığı veyahut üzerine analiz yapıldığı birtakım inceleme adımıdır. 1.2) **TASARIM**: Bu aşamada yazılım projemiz bir inşaat projesi gibi ele alınabilir. Öncelikle projenin tasarımı yapılır ve algoritmalar kurulur. Yazılımın fonksiyonelliği, işlevleri ve daha nice kurgusu bu bölümde gerçekleşir. Geliştirici kodlama evresine geçtiğinde kafasında herhangi belirsizlik olmaz. 1.3) **GERÇEKLEŞTİRİM**: Bu aşamada yazılım projesine ilişkin kodlamalar açığa çıkmaya başlar. Tasarım evresinde verilen kararlar ve tasarısı yapılan ortama uygun bir şekilde yazılım geliştirilir. Bir nevi mimari projesi çizilmiş olan inşaat, ustalar tarafından yapılmaya başlanmıştır. 1.4) **TEST**: Sistemin hayata geçtiği evredir. Yazılım hayat içinde bir yer edinir. Bir şirkette kullanılacak yazılımdan bahsediliyorsa şirketle ilgili birimlerin yazılımla ilgili eğitimlerin alınması, donanımlarının temin edilmesi, bağlantılı olduğu birimlerin buna entegre edilmesi, başka yazılımlarla bir entegrasyonu varsa bunun sağlanması ve veri tabanı bağlantıları gibi pek çok adım bu aşamada ele alınır. Veya söz konusu bir oyun yazılımı ise temel kullanıcı kitlesinin kullanımı için demolar veya versiyonlar çıkarılır. Proje artık yaşayan bir yazılım haline gelir. Yazılım kullanılır ve problemler ortaya çıkar. Problemlere karşılık yeni fikirler ortaya çıkar, yeni ihtiyaçlar oraya çıkar ve bunlar değerlendirilmeye alınır. Bu değerlendirmeler sonrasında bakım aşamasına geçilir. 1.5)**BAKIM**: Yazılım projesinin tesliminden sonra ortaya çıkan hata veya eksiklerin giderilmesi için sonradan yapılan düzenleme evresidir yazılımın türüne göre zaman alabilen bir aşamadır söz konusu yazılım eski bir yazılım ise bazen projenin kendisi kadar maliyet ve zaman alabilir. Günümüzde bu aşamaların doğrultusunda yazılım geliştirme aşamasının daha da gelişmesi açısından birtakım modeller kullanılmaktadır. Yazılım yaşam döngü modelleri adını verdiğimiz bu modellerin, süreç yönetimi ve takibi açısından şirketlerde hem yöneticiler hem çalışanlar için pek çok avantajı bulunmaktadır. Bu yazı içeriğinde bahsedilen modellerden: Şelale, Barok, Gelişigüzel, Big bang, V-Süreç, Spiral, Artımsal gerçeklik, Kodla-düzelt, Evrimsel gerçeklik ve Çevik modellerin birkaçına değinilmiştir.

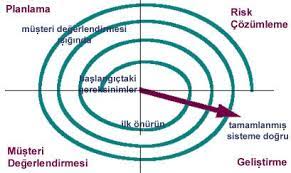
**ŞELALE (WATERFALL) MODELİ**:  
Şelale modeli, yinelemeli olamayan bir yapıya sahiptir ve kendisi yazılım mühendisliğinin temel modelidir.1968 yılında Almanya’da düzenlenen yazılım mühendisliği konferansında Dr. Winston W. Royce tarafından ilk defa tanıtılmıştır. Modelin aşamaları; **gereksinim, analiz, tasarım, kodlama, test ve entegrasyon, teslim ve bakımdır**. Safhalar, bir sonraki safhanın girdisi olduğundan geri dönüşlere izin verilmez ki bu tarz bir dönüş fazlasıyla maliyetli olur. Modelde her aşama bir önceki aşamaya bağımlıdır. Aşamalardan biri tamamlanmadan diğerine geçilmez. Aynı zamanda modelin her safhasında **belgeleme** gerçekleştirilir. Müşteri ile olan temas gereksinim safhasında etraflıca ve detaylı olarak alınır ve müşteri ile temas minimuma indirgenmek hedeflenir yani müşteri sürecin her aşamasına dahil olmak zorunda kalmaz. Modelin artıları bir yana oldukça dezavantajlı olduğu yönler bulunmaktadır. Modelde safhalar arası keskin sınırların olması ve geri dönüşlülük özelliğinin olmaması belki de modelin en belirgin eksikleri olabilir. Proje geliştirme sürecinde bazen safhalar arası dönüşler veya atlamalar gerçekleşebilir. Bu gibi durumlara elverişli olmayan şelale modeli bu sebepten ötürü kullanım açısından günümüzde büyük kayıp yaşamaktadır. Bunun yanında müşteri ile gereksinim aşamasında yapılmış olan detaylı analiz ve tek plan ileriye dönük sorunlar yaratabilir. Projede gerçekleşecek ufak değişiklikleri projeye entegre etmek oldukça fazla zaman ve paraya mâl olabilir.

**Şelale modelinde ortaya çıkabilecek sorunlar ve nedenleri**:   
1- Müşterinin tam olarak ne istediğini bilmemesi ve gereksinim aşamasında yapılan toplantılarda yeterli detayların alınamamış olması.   
2- Yazılım sürecin sonunda müşteriye teslim edilir bu demektir ki müşterinin yazılımı test etmesi sürecin sonunda olacaktır. Bunun sonucunda yazılımda oluşmuş aksaklıklar ve hatalar ile birlikte müşteri bu yazılımı kabullenmek durumundadır. Sonralarda yapılacak olan bakımlar hem müşteri hem de geliştiriciler için zahmetli olacaktır çünkü şelale modeli doğrusaldır ve yapılan değişiklikler tüm sistemi etkiler.  
**GELİŞİGÜZEL MODEL**:  
 Yazılım geliştirme sürecinde herhangi bir teknik veya modelin kullanılmadığı, yalnızca geliştiriciye bağımlı hatta bazı zamanlarda geliştiricinin kendisinin bile anlayamayacağı biçimde hazırlanmış proje biçimidir. Gelişigüzel model daha sonradan yapılacak düzenlenme ve bakımlara elverişli olmayan bir modeldir. Daha çok yazılıma yeni başlayanların iş yapma biçimine verilen isimdir.  
**BAROK MODELİ:**  
Daha çok 1970’li yıllarda kullanılmış olan bu yöntemi diğerlerinden ayıran en temel özelliği **belgelemeyi** ayrı bir aşama olarak ele almasıdır. Yazılım geliştirme ve test aşamalarından sonra belgelemeyi öngörür. Halbuki günümüz modellerinde belgeleme sürecin doğal bir parçası olarak gelişir. Modelde aşamalar arası geri dönüşlerin nasıl yapılacağı tanımlanmamıştır. Yazılım yaşam çevriminin temel adımları burada doğrusal bir biçimde gerçekleşir.  
**BİG-BANG MODELİ**:  
Belirli bir sürecin takip edilmediği ve ilerleyişin bir sisteme oturtulmadığı basit bir modeldir. Süreç yönetimi kolay ve herhangi katı bir süreç gerektirmeyen model, minimum bütçe ile yürütülebilir. İş sonucunda ihtiyaçları karşılayan veya eksik karşılayan projeler çıkabilir. Geliştirici gereksinim ve analiz evrelerini yaptıktan sonra yazılımını geliştirir ve kendi test eder bu yol ile kısa sürede proje çıkarmak hedeflenir.  
**V-SÜREÇ MODEL:**   
Bu model şelale modelindeki geliştirim ve test aşamalarının ne zaman yapılacağını belirterek daha detaylı ve anlamlı hâle getirir. Modelde V çubuğunun sol tarafı geliştirme aşamasını sağ tarafı ise test aşamasını belirtir. V-süreç model ayrıca hata bulunduğu zaman dönülecek aşamayı da söyler. Sağ taraftaki test aşamalarında bir hata bulunduğunda bulunan aşamanın yatay olarak karşısına denk gelen geliştirme aşamasına geri dönülür. Modelde temel çıktılar **mimari, kullanıcı, gerçekleştirim** olarak 3 aşamada incelenir



Resim 1.0 V-Süreç

**1)KULLANICIULLANICI MODELİODELİ**: Geliştirim aşamasında kullanıcı ile gerçekleşecek temasları tanımlar. Bu temaslar genele olarak gerçekleştirim veya test hakkında kullanıcı beklentileri hakkında gerçekleşir. Sistemin nasıl test edileceği bu aşamada planlanır.  
2)**MİMARİ MODEL**: sistem tasarımı ve oluşacak alt sistemlere ve tüm sisteme ilişkin oluşacak bütün test detaylarını içerir.  
3) **GERÇEKLEŞTİRİM MODELİ**: Yazılım modüllerinin kodlanması ve test edilmesi işlemlerinin gerçekleştiği bölümdür.  
**V-SÜREÇ MODELİN AVANTAJLARI:** Belirsizlikler az ve tanımlamalar açıkça yapıldığı için BT (bilişim teknolojileri) projeleri için uygundur. Geliştirme sürecinde kullanıcı geri bildirimleri sonucu kullanıcının projedeki katkısını arttırır.

**SPİRAL MODEL**:  
Spiral yazılım geliştirme modeli temel olarak 4 ana bölüm içerir. Bunlar: **Planlama, risk yönetimi, üretim**, **kullanıcı yorumları.** Spiral model, **risk yönetimi** ve **prototipleme** üzerine kurulu bir modeldir. Her döngü aslında bir fazdır. İçinde bulunulan fazın risk analizi yapılarak o faz için planlanmış prototip geliştirilir Her döngünün sonunda yeniden planlama yapılarak alternatifler ve kısıtlamalara belirlenir. Bu model önceden geliştirilmiş yazılım parçalarının kullanıldığı projeler için oldukça uygundur. **PLANLAMA**: Üretilen ara ürün için planlama, amaç belirleme ve önceki ara ürün ile birleştirme. **RİSK YÖNETİMİ:** Risk durumlarının belirlenmesi. **ÜRETİM:** Ara (prototip) ürünün üretilmesi. **KULLANICI DEĞERLENDİRMESİ:** Ara ürünlerin kullanıcılar tarafından test edilmesi. 

**Resim1.1** Spiral model

**Spiral modelin avantajları**: Geliştirme sürecinde kullanıcının ara ürün testinde verdiği katkılar ve sürece dahil olması ile ileride yazılımı kullanacak olan kişilerinde ürünün geliştirilmesi aşamasında yer alması, ileride meydana gelebilecek olumsuz durumlarda tespit kolaylığı sağlar. Dahası yöneticiler ve çalışanlar iç içe olduğundan yöneticilerin süreci takip etmesi kolaylaşır.  
Geliştiriciler açısından ise yazılımın kodlanması ve test aşamalarının daha erken başlaması sağlanır bu sayede zamandan kazanç sağlanır. Bunların yanından Spiral modelin en büyük getirisi, risk yönetimi sayesinde zaman ve maliyet değerlerinin kolay tahmin edilebilir olmasıdır. Projenin kaliteye yönelik hedeflerinin de önceden belirlenmesinden sonra her döngüde alternatifler ve kısıtlamaların belirlenmesi ile hedeflerin belirlenmesi diğer modellere göre daha kolay. Bunca faydasının yanında modelin küçük projelerde kullanılmaya elverişli olmaması, uygulanmak için tecrübe gereksiniminin olması, risk yönetimi olgusuna dayandığından alt yüklenici kullanımında zorluklar yaşanması gibi dezavantajları da bulunmaktadır.   
**ARTIMSAL GELİŞTİRME SÜREÇ MODEL:**  
Modelde sürüm kavramı ön plandadır. Üretilen ve uygulamaya alınan her ürün sürümü, birbirini içerecek biçimde kümülatif olarak ilerleyen bir sürüm zinciri oluşturur. Giderek artan bir işlev kümesi oluşur. Öncelikle ürüne ait kemik bir yapı oluşturulup daha sonra sürümler ile işlevsellikler kazandırılarak yeni sürümler elde edilmektedir. Modelde proje parçalara bölünür ve kullanıcı kitlesinin önceliğine göre parçalar sıralanır. Sıralama sonucunda birer ara ürün çıkarılır ve bu ara ürünler kullanıcılar tarafından bir nevi testten geçer. Yani model üretilmeye devam ederken aynı zamanda kullanılmaya da devam eder. Model uzun soluklu projeler ve eksik işlevselliğin göz ardı edilebileceği projelerde kullanılmaya uygundur. Bu modelde kullanıcı ile geliştirici iç içe geçtiğinden başarısız olma riski daha azdır. Öte yandan bir ara üründe değişiklik yapılabilmesi veya hata giderilebilmesi için yeni sürüm beklenmek zorundadır.  
**KODLA VE DÜZELT**:  
Çoğunlukla bir işe yeni başlandığında veya bir problemi çözmek için kullanılabilecek en basit yöntemdir. **En basit yazılım geliştirme (Cowboy Coding)** olarak bilinen bu modelde en kısa sürede prototipleme yapılarak sonuca gitmek hedeflenir. Planlama analiz gibi aşamalar üzerine düşülmeden temel probleme odaklanılır. Küçük projelerde kullanılmaya uygundur ve emeklilik safhası vardır. Bakımı az ama zordur.  
**EVRİMSEL GELİŞTİRME MODELİ:**  
Model ilk tam ölçekli modeldir. Prototipleme ön plandadır. Evrelerin başarısı kendinden önceki evrenin başarısına bağımlıdır. Büyük çaplı firma, fabrika yerleşkeleri için uygundur. Üretildikleri yerler için tam fonksiyonellik içerirler. Model değişiklik ve bakım açısından zorluklar çıkarabilmektedir.  
**ÇEVİK (AGİLE) YÖNTEMLER:**  
Çevik yöntemler yazılım geliştirme aşamasındaki verimi, başarı oranını ve ekip içi iletişimi arttırmayı amaçlayan bir yazılım geliştirme süreç modelidir. Başlıca en çok kullanılan Çevik yöntemler: **SCRUM, XP (extreme programming)** yöntemleridir.

**SCRUM**: İsmini Amerikan futbolundaki bir oyun taktiği olan tüm takımın birlikte hareket ettiği bir yöntemden alır. SCRUM da büyük projeler sprint adı verilen küçük parçalara ayrılarak ayrı ayrı geliştirilir. SCRUM da asıl önemli olan ekip içi uyumdur. Bu bağlamda SCRUM toplantıları tarzı etkinlikler ile ekip bağları geliştirilir. SCRUM 3 temel kavramdan temel alır. Bunlar: Roller, Toplantılar, Bileşenlerdir. **ROLLER**: Proje ekibi, ürün sahibi, SCRUM yöneticisi ve SCRUM takımından oluşur. Ürün sahibi projeye genel hatlarıyla hâkim olan projenin beynidir. SCRUM yöneticisi projenin gidişatını ve SCRUM ekibini denetler. SCRUM ekibi yazılım geliştiricilerden oluşmuş birbirleri ile iletişim halindeki gruptur. **TOPLANTILAR**: Toplantılar SCRUM için olmazsa olmazdır. Her gün yapılan bu toplantılarda projenin gidişatı, önceki gün ne yapıldığı ve bugün ne yapılacağı, projeden çıkan hatalar üzerine konuşulur ve içinde bulunulan sprint için değerlendirme yapılır. **BİLEŞENLER:** proje için gereksinim dokümanı oluşturulur ve projede yapılacak olanlar bu dokümana basitçe yazılır. Gereksinim dokümanına bağlı olarak sprint dokümanları oluşturulur ve amaç her sprint dokümanının gereksinim dokümanına uygun hazırlanmasıdır.   
SCRUM; paradan ve zamandan tasarruf sağlaması, karmaşık görünen detaylandırılmamış projelerdeki kullanım kolaylığı, değişen gereksinimlere hızlı tepki vermesinin yanında son teknolojilere de kolay adapte olması gibi yönleri ile günümüzde en çok kullanılan yazılım hatta bazı zamanlarda sistem geliştirme biçimi olmuştur. Kullanıcı ile sürekli bir geri bildirim ağı içerisinde olması projede gerçekleşebilecek hataları minimuma indirir. Ayrıca diğer metodolojiler gibi yinelemeli olması ve böl ve fethet mantığı üzerine kurulu yapısı da yine kendisini alternatiflerinden öne çıkartıyor.

**EXTREME PROGRAMMİNG(XP):**

1996 yılında Kent Beck ve ekibi tarafından oluşturulmuş olan bu model, 4 ana maddeden oluşur. Basitlik, cesaret, geri dönüşlülük ve iletişim bu maddelerdir. **BASİTLİK**: Yazılan kodun yalın, karmaşıklıktan uzak ve anlaşılır olması gerekir. Dokümantasyonun uzun olmasından kaçınılır. **CESARET**: Projede önyargılardan arınılıp korkusuzca kod üzerinde işlem yapılmalıdır. Bazen yapılan kodun üzerinde değişiklikler yapmak yeniden kod yazmaktan daha uzun olabilir bu bizim için hem daha maliyetli hem daha fazla zaman kaybı demektir. Bu gibi bir durumda koda manevi bir şekilde bağlanmak yerine yeri geldiğinde kodu tamamen çöpe atmaktan çekinmemeliyiz. **GERİ DÖNÜŞ**: Yazılımın geliştirilmesi aşamasında proje üzerinde geri dönüş ve tekrar kontrollerin yapılması ile kodda bulunan hatalar azaltılır ve kaliteli bir ürün ortaya çıkar. Bunun için de müşteri ve geliştiriciler iletişim içinde olmalıdır. **İLETİŞİM**: İletişim yukarıda da bahsedildiği gibi yazılım geliştirme süreci için çok önemli bir rol oynamaktadır kaliteli içeriklerin üretilmesi için kullanıcı yorumları ve test sonuçları geliştiriciler için yol haritası oluşturabilir. Bunun yanında sadece müşteri-geliştirici iletişimi değil geliştirici-geliştirici iletişimi de son derece önemlidir.   
**ÇEVİK YAKLAŞIMLAR VE ŞELALE MODELİNİN KARŞILAŞTIRMASI**:  
Her iki yaklaşım biçimi de projelerin başarısını arttırmak gayesi üzerine ortaya çıkmış olsa da bir noktadan sonra şelale modeli çevik yaklaşımların tam zıttı pozisyonda bulunuyor. Kesiştikleri ve ayrıldıkları noktaları maddeleyecek olursak:

* İki yaklaşımda da planlama söz konusudur fakat şelale modelinde uzun süreli ve titizlikle hazırlanan planlama, çevik yaklaşımlarda planlamalar kısa vadeli hazırlanır.
* Şelale modelinde bir görev sıralaması bulunur. Bir safha bitmeden diğerine geçiş yapılamaz ve safhalar arası geçişler tanımlanmamıştır. Buna karşın çevik yaklaşımlarda bu tarz bir düzen söz konusu değildir. Sondan başlama bile mümkündür.
* İki modelde de detaylı inceleme vardır. Şelale modelinde incelemeler dokümantasyonlar ile sağlanırken çevik yaklaşımlarda incelemeler yeni her yineleme ile birlikte kullanıcı değerlendirmeleri vasıtasıyla yapılır.
* Kullanım kolaylığı avantajı olmasına rağmen şelale modelinde geri dönüşlerin tanımlı olmaması maliyeti oldukça arttırır. Çevik yaklaşımlarda değişimler, geri dönüşler sürecin doğal parçası gibidir ve yazılımın nihai hale gelmesini sağlayan etkilerdir.
* Çevik yaklaşımlar esnek yapısı ile yaratıcılık ve estetik anlamında bizlere olanaklar sağlar iken şelale modelinde katı bir mühendislik söz konusudur.
* Detaylı bir planlamanın bulunduğu şelale modeli; çok büyük, değişme ihtimali bulunmayan, tam anlaşılmış projeler için uygundur. Çevik modeller ise tam zıttı şekilde çalışan projeler için uygundur.

**MEDYA BAĞLANTILARI VE KAYNAKÇA:**

Medium:<https://medium.com/@dtemel844>

LinkedIn: [www.linkedin.com/in/delil-temel-17aa6a222](http://www.linkedin.com/in/delil-temel-17aa6a222)

Github: <https://github.com/Delilt>

[www.mikra.com.tr/nasil-calisiriz/yazilim-gelistirme-sureci](http://www.mikra.com.tr/nasil-calisiriz/yazilim-gelistirme-sureci)

<https://medium.com/@omerharuncetin/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-543c7879a742>

Resim1.0: https://www.google.com/url?sa=i&url=http%3A%2F%2Fw3.bilecik.edu.tr%2Fbilgisayar%2Fwp-content%2Fuploads%2Fsites%2F75%2F2019%2F02%2Fders-2.pdf&psig=AOvVaw3MRvd-ELPBFlgBxDTiIr7E&ust=1647775621202000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCKjwuvSI0vYCFQAAAAAdAAAAABAD

Resim1.1: <https://www.google.com/url?sa=i&url=https%3A%2F%2Fherturbilgi.com%2Fyazilim-sureci-modelleri%2F&psig=AOvVaw2NgjWHph0YwyGpzRHRAwlj&ust=1647775541169000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCPju1rWI0vYCFQAAAAAdAAAAABAE>

<https://tr.wikipedia.org/wiki/Yaz%C4%B1l%C4%B1m_ya%C5%9Fam_d%C3%B6ng%C3%BCs%C3%BC>

Doç. Dr. Deniz Kılınç Yazılım mühendisliği temelleri ders notları

Okan Öztürkmenoğlu Bilgisayar sistemleri temelleri ders notları

<https://tr.linkedin.com/pulse/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-nedir-veysel-ugur-kizmaz>

<https://osmanozaydin.com/yazilim-yasam-dongusu-ve-agile-yazilim-gelistirme/>

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

<https://scholar.google.com.tr/scholar?q=art%C4%B1msal+geli%C5%9Ftirme+s%C3%BCre%C3%A7+modeli&hl=tr&as_sdt=0&as_vis=1&oi=scholart>

https://makersturkiye.com/cevik-agile-yaklasimlar-mi-selale-waterfall-modeli-mi/#:~:text=%C3%87evik%20Yakla%C5%9F%C4%B1m%20vs%20%C5%9Eelale%20Modeli&text=Her%20ikisinde%20de%20bir%20planlama,etti%C4%9Fi%20bir%20g%C3%B6rev%20s%C4%B1ras%C4%B1%20vard%C4%B1r.